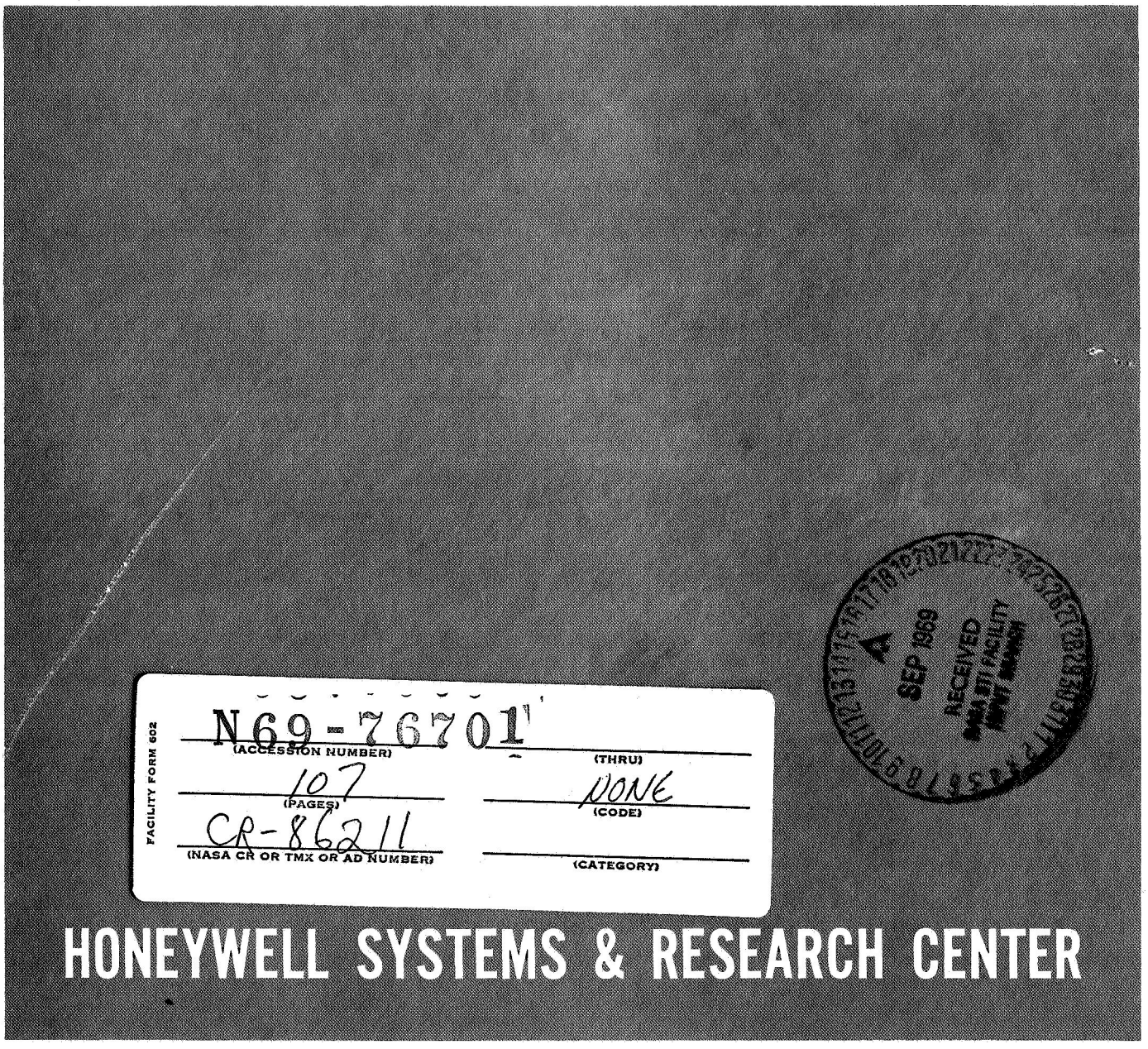
~~CR-86211~~
CR-86211

Honeywell Document 12017-IR2

Phase II - Interim Report

SPACEBORNE MEMORY ORGANIZATION ASSOCIATIVE DATA ACQUISITION SYSTEM

Contract No. NAS 12-38



12017-IR2 September 1966

Phase II - Interim Report

SPACEBORNE MEMORY ORGANIZATION ASSOCIATIVE DATA ACQUISITION SYSTEM

by

D. C. Gunderson
C. W. Hastings
H. R. Holt

September 1966

Prepared Under Contract No. NAS 12-38

Honeywell Inc.
Systems and Research Center
Minneapolis, Minnesota

CONTENTS

CONTENTS (Continued)

ILLUSTRATIONS

SECTION I
INTRODUCTION

This report contains the results of work performed on Phase II of Contract NAS 12-38, Spaceborne Memory Organization, during the period April 1966 through September 1966.

Phase II of the contract is aimed at the design of a space vehicle data acquisition system based on the use of associative memory techniques. The Phase II effort is divided into three major tasks: definition of space vehicle data acquisition requirements, system design of an associative data acquisition system, and evaluation of certain modified data compression algorithms by simulation. The first task has been completed, the second is partially completed, and the third is presently being initiated.

The spaceborne data acquisition system requirements are defined in Section II. Considerable time has been spent attempting to develop some "feel" for those characteristics of space vehicle instruments which are of interest in computer-controlled instrumentation systems. It was originally hoped that a classification scheme for these instruments could be developed which would provide a reliable gauge of the conversion and computation rates which will be required in 1975-vintage systems, but it was found that space missions and space vehicle instruments are too heterogenous for any such scheme to be developed in the time available. Nevertheless, enough was learned to allow making choices for essential system design parameters (such as the number of "strands" of the associative computer, the number of bits required to represent a single datum, etc.) which are believed to be acceptably realistic.

The organization of the associative data acquisition system is described in Section III. The approach requires one word (strand) of associative computer for each data source in the system. The associative computer is under the supervision of a general-purpose computer which provides overall system control and serves as a program memory for the associative computer. The associative computer performs the functions of multiplexing, analog-to-digital conversion, and data compression. The major advantages of this system can be directly attributed to the highly parallel structure of the associative computer. While the major part of the system design effort is completed, in a number of areas alternate approaches are still being evaluated; these alternate schemes are described in this report. One scheme in each area will subsequently be selected for the final system design.

The final task in the project is to evaluate a modified method of using otherwise conventional data compression algorithms. The search capabilities of the associative computer allow certain numerical results, obtained by applying data compression algorithms to all data sources, to be evaluated all jointly rather than each independently as in the conventional usage. A statement of the objectives and main steps of this task is given in subsection V-C of this document.

SECTION II
SPACE VEHICLE DATA HANDLING REQUIREMENTS

A. WHY STUDY THE INSTRUMENTS?

The design of digital computer systems is often quite dependent on essential dimensions of the system; the size of a core memory or the word length in a scientific computer, the number of bits to which the input variables are expressed in special-purpose fast multipliers, and the required internal alphanumeric character format in a business-system computer are examples of very sensitive dimensions. Similarly, the design of an associative computer is sensitive to the number of data fields (pieces of information) stored in each "word" (hereafter called "strand"), the number of bits required to express each data field, and the total number of strands. The latter dimension, in the type of system considered here, depends directly on the number of instruments and status-word sources ("data sources") carried aboard the space vehicle.

The design tradeoffs in a complete digital system design are complicated, and many critical decisions (whose effects are thereafter pervasive) depend on tradeoffs which are in turn dictated by these dimensions; for example, a random-access core memory of some given bit capacity might most naturally be operated in a coincident-current mode if one choice was made of word length and number of words, whereas a linear-select mode might be preferable if the word length were increased by a factor of four and the number of words were decreased by a factor of four. In anticipation that situations of this sort might come up in the design of the associative computer, the program for Phase II of this contract called for an initial study in some detail of the characteristics of typical space vehicle instrumentation systems, in order that the critical dimensions for the associative computer could be chosen with as much realism as possible.

This study did not achieve everything hoped for, but it did yield enough "feel" for the problem that the dimensions finally chosen - 256 strands, 192 bits of storage per strand - are believed applicable to a fair number of future space vehicle missions. The 192 bits depends mostly on the data compression algorithms used; if the standard input conversion accuracy of all analog and many digital inputs is 10 bits, two extra numeric bits are added in memory to prevent roundoff error from affecting significant figures, and the data compression algorithms used require the storage of 15 data fields per strand, that accounts for 180 bits per strand with 12 bits left over for storing control (see Subsection III-B) information. (A sign bit may not be necessary during the computation.) Whereas the number of instruments aboard a space vehicle may range from half a dozen to several hundred, and the approach taken here requires one strand for each instrument, it appears that 200-300 is a generous amount for most missions; thus the choice of 256, which is a power of 2 (2^8) and hence is a desirable computer dimension. It is doubtful if this dimension is likely to exceed $2^{10} = 1024$ for any mission now easily foreseeable, or drop below $2^5 = 32$ for any mission for which the associative computer technical approach is really appropriate. This dimensional range is still a bit wide, but is narrow enough to permit few trenchant generalizations regarding circuit techniques and computer organization.

The most striking illustration of the dimension-dependence of the associative computer design is probably the design approach to the selection and detection unit, described in Subsection III-D. The same dependency crops up in each of the three competing design approaches described there. However, even the choice of memory components may also depend on these critical dimensions. Thus, the study of instruments gave valuable qualitative information although it did not yield the ultimate procedure for translating space mission requirements into data processing system specifications.

B. INSTRUMENTS AND EXPERIMENTS

1. Space Exploration Data Processing

The viewpoints expressed in this section are those of a computer designer without prior expertise in space systems engineering, interested in the properties that space vehicle instruments exhibit with respect to computer tie-in. Very little material has apparently been written from this point of view. The following references were found to be helpful in a qualitative way:

(1) Manned Mars Surface Operations, AVCO/RAD, Wilmington, Massachusetts; AVCO/RAD-TR-65-26, Contract No. NAS 8-11353, 30 September 1965.

(2) Space Measurements Survey for DASA, Electro-Optical Systems, Inc., Pasadena, California, Volume I, 1 July 1962; Volume II, 28 February 1965. EOS Inc., Report No. 1890, DASA Report No. 1277, WEB No. 07.013, Contract Number DA-49-146-XZ-100. Volumes I and II have since been issued, along with additional material intended to comprise a Volume III, as a NASA Special Publication, Instruments and Spacecraft, edited by H. L. Richter, Jr., NASA SP-3028, NASA Scientific and Technical Information Division, Washington, D. C., 1966.

(3) NASA PATTERN Relevance Guide - Volume III, Honeywell Aeronautical Division, St. Petersburg, Florida; Honeywell Document No. 8-20207-RG, Contract No. NAS 8-20207, 15 October 1965.

(4) Small Standard Satellite (S^3) Feasibility Study, NASA Goddard Space Flight Center, Greenbelt, Maryland; Goddard Document No. X-724-66-120, March 1966.

(5) A Study to Determine an Efficient Data Format and Data System for a Lightweight Deep Space Probe, Research Report No. 1, TRW Systems, TRW, Inc., Redondo Beach, California, Contract No. NAS 2-3254, 18 February 1966.

2. Types of Space Exploration Missions

There is no particular unifying principle which one can state about all space vehicle instruments, any more than there is about all instruments in general; space vehicle instruments are a very heterogeneous lot. They exhibit a profuse variety of purposes, operating modes, output formats, and idiosyncrasies.

There are many possible types of space missions. Space vehicles may reasonably be subdivided into the following categories: space probes, flyby probes, orbiters, impacters, landers, lander-rovers, rovers, shelters. In "systematic planetology" (planetary exploration), one may distinguish the following more or less spherically concentric separate zones of exploration for our own planet, and most of them would apply equally to any other planet: endosphere (core); lithosphere (rocky crust); hydrosphere (oceans, lakes, icecaps, etc.); biosphere (life forms); atmosphere; electro-magnetosphere (belts of charged particles); gravisphere (region of significant gravitational influence); and solar system (everything else, for the time being, until the first interstellar probe goes up).

Vehicles of the same type may still be used for widely varying purposes. An orbiter, for instance, may be intended to map the planet surface, to predict weather by observing the cloud cover, to investigate gravitational anomalies, or to investigate the density and nature of charged particles at various altitudes.

The order in which the various types of space vehicles are listed is approximately that in which they would be used to study a previously unvisited planet. Also, the longer-range vehicles are more apt to be multiple-purpose; it is now easy enough to put up earth orbiters that some of them are single-purpose, but it would not be reasonable in the near future to plan on using a Mars orbiter for just one experiment or class of experiments.

The data rates attainable between a space vehicle and an earthside tracking station fall off roughly according to the inverse-square law. Hence, there are many decimal orders of magnitude of difference between the data rates of earth orbiters (100,000 or more bits per second in some cases) and those of long-range probes (sometimes as small as 1 to 5 bits per second).

3. Classification of Instruments

The following information comprises roughly what a computer design engineer would like to know about the complement of instruments aboard a space vehicle:

(a) The form of the output reading (digital, analog, random pulse, synchronous pulse, etc.).

(b) The equivalent binary precision (how many bits are needed to adequately represent the reading).

(c) The minimum sampling frequency (the inverse of how long the instrument may be left unread).

(d) The mode of operation (continuous, continuous at certain times, intermittent, occasional, one-shot).

(e) The delay (if any) between the receipt by the instrument of a command to supply a reading and the availability of that reading.

(f) Any relevant special characteristics (such as "monotonicity" - certain devices present their readings as counts, and these counts usually always increase monotonically and never decrease).

Also, a space systems engineer would like to know one more thing;

(g) How much digital or analog electronics of a basically computational nature has been included in the "instrument" package, and could be eliminated at a saving in size, weight, and power consumption now that the instrument is being connected to a full-scale associative computer or GP computer.

The precision and sampling frequencies could if necessary be given in terms of ranges without too much loss of information. The computer design engineer does not necessarily even need to know what the instruments are, but only needs to know their characteristics when they are viewed as data sources, and can as well treat them collectively as one-by-one. Thus it may suffice to state, for instance, that there are X analog instruments whose outputs must be converted to 10-bit precision and which must be sampled at least every 20 milliseconds, and so forth. One useful approach, taken by AVCO/RAD in the first reference given above, is to treat instruments for far-future missions in terms of the basic properties of the measurements to be taken rather than of the instruments now in existence, since whatever instruments are invented to take those measurements will have to tailor their precision, sampling frequency, and so forth to whatever is appropriate for the measurement.

Gathering this sort of information about space vehicle instruments proved to be a difficult task. The difficulties which stand in the way include the following:

(1) The development of suitable space-vehicle instruments sometimes lags the formulation of mission requirements by several years, and radical departures from contemporary instrumentation are often called for. Hence, the characteristics of many space-vehicle instruments are not predictable with much confidence over

the time period 1975-1985, particularly where there are competing instruments with radically different data formats (the spectrograph and the chromatograph for atmospheric composition analysis, for instance).

(2) The capabilities and needs of space-vehicle data processing systems have not discernibly influenced the design of space-vehicle instruments yet, and such influence would be desirable. When an instrument is to be sampled by a computer capable of data compression, the most convenient type of instrument is one which can be sampled without a significant delay at an arbitrary time (i. e., "continuously"), and which normally produces an output consisting of one or two pieces of numeric data of the precision (i. e., "word length") taken as standard for the computer. It makes no sense in this context to speak of how many "bits per second" the instrument produces, since that depends on how often it is sampled; yet this figure is the only data-processing-type parameter given in many reports for most instruments, and the inference is that a sampling rate has been chosen in advance (and possibly even in some way built into the instrument) to provide for a minimum loading of the space-to-ground communications system consistent with adequate earthside reconstruction of the time history of the variable read by the instrument. If now it is assumed that a computer capable of data compression is available aboard the space vehicle, such an advance choice becomes not only unnecessary but undesirable; the choice should instead by made adaptively in real time by the computer. In some cases, these considerations may mean some simplification of the instrument; in others, they would favor one type of instrument (such as chromatograph) over another (such as a spectrograph) because the first adheres to a standard data format and the second does not.

(3) The data rates for long-range space exploration vehicles are often assumed to be very low (8-10 bits per second). On the other hand, the data rates for satellites in orbit about the earth are often quite high (many kilobits per second). There is a strong motivation for using data compression in each case: primarily to conserve the power needed for data transmission in the long-range case, and primarily to avoid overwhelming the earthside computer center with masses of redundant data in the earth-orbiter case. Also, one might reasonably expect that the assumed data rates for the exploratory vehicles assigned to any given celestial body will increase with time, as the nature of the exploration of that body changes from an initial contact to a detailed mapping and examination. Hence, it is not at all reasonable to assume a "typical" space vehicle data rate, since the variation is too great. It may be reasonable, however, to choose as "representative" some rate characteristic of a vehicle doing a detailed study of, say, the Martian surface.

The following classification scheme for space vehicle instruments is much weaker than the first one discussed but it still is of some value to the computer designer:

(A) High-speed or variable-speed continuously readable data sources

(B) Low-speed continuously readable data sources

(C) Data sources which can be read only after a certain delay following their receipt of a command

(D) Data sources corresponding to experiments which are only done once or a few times during the mission

(E) Anomalous data sources which present a non-numeric interface

All instruments whose characteristics have been studied appear to fit into one of the classifications (A) through (E). Obviously there are many possible subcategories, of which the following is a non-exhaustive list:

(A. 1) Raster-Type Instruments

- (A. 1. 1) Television camera
- (A. 1. 2) Mapping radar
- (A. 1. 3) Scanning radiometer

(A. 2) Counter-Type Instruments

- (A. 2. 1) Omnidirectional particle counters
- (A. 2. 2) Directional particle counters ("Geiger telescopes")
- (A. 2. 3) Frequency interval timer
- (A. 2. 4) Laser gyroscopes

(B. 1) Solid-State Sensors

- (B. 1. 1) Thermocouples
- (B. 1. 2) Piezoelectric strain gages
- (B. 1. 3) Piezoelectric manometers
- (B. 1. 4) Photocells
- (B. 1. 5) Hall-effect or magnetoresistance magnetic field probes
- (B. 1. 6) Resistance thermometers

(B. 2) Shaft-Position Encoders (angular or longitudinal position)

- (B. 2. 1) Brush contact
- (B. 2. 2) Photoelectric
- (B. 2. 3) Magnetic . . .

Instruments of classes A and B would be suitable for direct connection to an associative computer. However, all of them might not be connected to the same associative computer, because of the vast difference in sampling rates between, for example, a television camera and a thermocouple which indicates the temperature of the computer memory container. In general, connecting a number of instruments of varying minimum sampling frequencies to the same associative computer will result in "oversampling" some of them according to the conventional sampling theory view; but the data compression algorithms will automatically get rid of any extra redundancy introduced in this manner.

Instruments of classes (C) and (D), referred to elsewhere in this report as "on-demand" instruments, require a fundamentally different sort of servicing by the data processing system than do those of classes (A) and (B). These instruments need to be activated and read at certain well-defined and isolated times, and hence for them the associative computer functions as an "alarm clock" featuring multiple future alarm settings as is described in Subsection IV-B.

Instruments of class E could be connected to the associative computer, but it would not be particularly profitable to do so. A reading from a class (E) instrument would probably just have to be transmitted intact to the GP computer, or else processed in a mode unique to that instrument while most of the associative computer strands did nothing, and hence it is just as well to connect such instruments directly to the GP computer to begin with. An inexpensive way of connecting a number of them, if none of them has a really high data output rate, is the "roll-call" approach: a large number of instruments are connected to one GP computer input/output data bus, and each one has a unique "roll-call address." An instrument's connection interface logic responds to signals on the bus only when the GP computer's "external function" control lines are supplying a command containing its roll-call address. Hence, if the GP computer is programmed properly, data and commands passing to or from one instrument do not interfere with those passing to or from another.

Certain instruments can reasonably be connected to the associative computer, but their output readings require special treatment because they consist of multiple pieces of data. For instance, a "scintillator" produces a reading consisting of a pair of data (ΔE) and ($E - \Delta E$); the second of these is usually plotted versus the first, rather than plotting either of them versus time.

Also, sometimes the output of several instruments can be processed to yield an estimate of the output of some other instrument, for calibration and verification purposes. Such verification computations would fall to the GP computer to perform, rather than to the associative computer.

4. What "Well-Behaved" Instruments Look Like

The ideal behavior of an instrument from the point of view of a computer, as has already been stated in slightly different words, is roughly as follows:

(1) The instrument is continuously readable – a reading is always there ready to be taken.

(2) There is no delay within the instrument for reading – when the computer attempts to obtain the reading, it is available instantly.

(3) The reading conforms to the same precision and sampling frequency requirements as a large number of other readings.

(4) The same output data lines of the instrument always have the same information – the instrument does not itself "multiplex" or "subcommutate" different types of information.

(5) The instrument changes at a fairly steady rate, rather than alternating periods of furious activity with periods of quiescence.

Obviously the characteristics of instruments must be determined by factors which override the wishful thinking of the computer designer, and so many instruments will always be included whose behavior flagrantly violates the above rules. Nevertheless, there are a few cases where there may be competing instruments which vary considerably in how "well-behaved" they are; for instance, a gas chromatograph would probably be easier to connect to an associative computer than a spectrograph.

5. Processing Objectives

The tasks to be performed by the space vehicle data processing system are broken down in detail in Subsection II-C. The purposes of all this processing may be summarized, in an admittedly oversimplified manner, as follows:

(A) Data Acquisition

- (A. 1) Collection
- (A. 2) Conversion
- (A. 3) Formatting

(B) Data Conditioning

- (B. 1) Normalization
- (B. 2) Linearization
- (B. 3) Averaging (for noise elimination)
- (B. 4) Filtering (of many different kinds)
- (B. 5) Wild point rejection

(C) Data Compression

- (C. 1) Redundancy removal
- (C. 2) Averaging (for reducing the number of bits to be transmitted)

(C. 3) Truncation of word length

(C. 4) Fixed-point-to-floating-point conversion

(C. 5) Encoding

(C. 6) Statistical extraction of data features

Figure 2 of Reference 2 indicates how very many schemes of data compression there are. All of them have the same fundamental objective of reducing the number of bits which must be transmitted back to earth, but some have other side effects also – good and bad. At their best, they improve the interpretability of the information to the experimenter; at their worst, they render the data meaningless.

Data compression schemes differ from one another so greatly that it is not always obvious that all of them even have the same main objective. Different data compression algorithms are sometimes treated in the literature simply as different basic types of processing which have to go on aboard a space vehicle.

6. Special Requirements for Spinning Satellites

Spinning satellites are commonly used for certain types of missions, since they are easy to stabilize, and since the vehicle itself then automatically causes certain instruments to scan which otherwise frequently need to be rotated. The spin rate of the vehicle changes very slowly, and an "origin" signal can be produced by solar or horizon sensors in various ways to indicate to the data processing system when a complete spin resolution has occurred. Hence, the spinning satellite is an efficient means of providing a predictable rotating instrument platform.

It falls to the data processing system of the satellite, however, to cope with certain complications which result from the fact that the satellite is spinning. For one thing, the period of spin is very unlikely to be synchronized with the satellite on-board time clock, and data acquisition may have to be performed in time with either the spin rate or the clock. For another thing, the output of certain instruments (notably "Geiger telescopes") must now be interpreted as if it were "multiplexed" or "subcommutated," since when the angle of the instrument's axis is continually changing with respect to the horizontal, the reading which it presents at time t may not have the same physical significance as that which it presents at t + 10 milliseconds. For instance, the flux of charged particles at a given altitude, latitude, and longitude is still a function of the angle the trajectory of the particles makes with the horizontal.

Typically, the experimenter divides up one complete circular arc into 64 equal intervals (of about 5-1/2 degrees of arc each), and sets up the experiment so that the data values then correspond to these arc prints and to no others during a satellite spin revolution. To complicate matters further, the data processing system should capitalize on any data symmetries around this circular arc and average the corresponding data points; for instance, the readings taken at 11 degrees, 169 degrees, 191 degrees, and 349 degrees (assuming that 0 degrees here means straight vertical) may all have the same physical significance if the spin axis of the satellite is exactly perpendicular to the local magnetic field. The address modification required by this particular averaging task is tricky enough that the task should probably be left to the GP computer, unless there are enough similar instruments to warrant programming the associative computer to carry out the operation for all of them at once. One other possible approach here is hardware demultiplexing the output of one such spinning-satellite instrument to 16 (or 32, or 64) different strands; this approach might be worthwhile in some cases, but adds a certain amount of size, weight, and power consumption.

C. DATA SYSTEM DESIGN PHILOSOPHY

Figure 2-1 shows a conventional space vehicle data acquisition system, and may serve as a rough functional schematic for any data acquisition system. The functions performed by all subsystems of the system shown in Figure 2-1 must always be taken care of somehow in any data acquisition system, although their specific arrangement may vary. These functions are described in the following paragraphs, along with various ways of handling them corresponding to (1) a conventional data acquisition system with an analog multiplexer, (2) a conventional data acquisition system with a digital multiplexer, (3) a data acquisition system organized around an associative computer. The purpose of this discussion is to state certain definitions and premises, and to show that the associative computer approach is a very natural one to take.

(1) Data Sources – There are always a number of data sources, some of which need to be sampled oftener than others. It is customary that all data sources in a system use a standard interface, or at least one of several standard interfaces; a standard interface specification for analog data sources might, for instance, be that each reading present as a signal of from 0 to E volts, with 0 representing the lowest point $y_{i_{min}}$ of the physical variable y_i being sensed and E representing the highest point $y_{i_{max}}$. $y_{i_{min}}$ is called the offset, $y_{i_{max}}$ is called the full-scale value, $(y_{i_{max}} - y_{i_{min}})$ is called the range, and $\frac{E}{(y_{i_{max}} - y_{i_{min}})}$ is called the scale factor for the i^{th} data source.

Inherently analog data sources include thermocouples, piezo-electric manometers and strain gates, electrostatic and other rotor gyros, Hall-effect and magnetoresistance magnetic field probes, and so on. There are also many common data sources which produce inherently digital outputs; these include devices

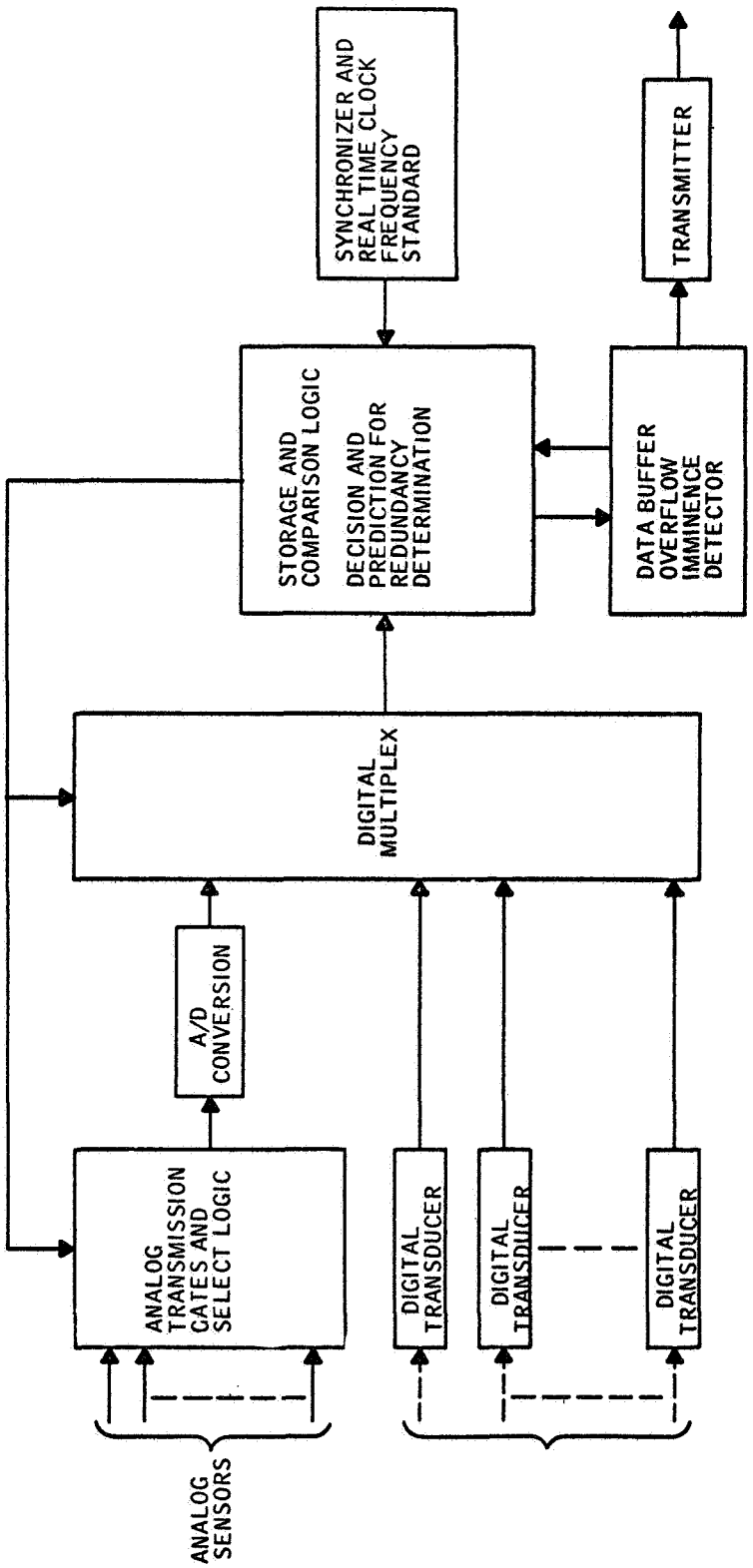


Figure 2-1. Conventional Space Vehicle Data Acquisition System

which present a reading via a digital counter (particle counters, interval timers, laser gyros, and so on), and even some physical devices such as interferometers.. Moreover, there are some cases where a quantity is best defined in terms of a linear or angular physical displacement of a shaft; such a quantity may be read out digitally in "Gray code" using a code strip or code wheel* with contact sensing or else photoelectric or magnetic pickoff, but it may equally well be read out as a voltage by the use of some analog pickoff device such as a slide wire. Thus, in a typical space vehicle (or other) data acquisition system there will be some data sources whose outputs are naturally analog, some whose outputs are naturally digital, and some where there is a choice; the system must tie in each type of data source in the most efficient, reliable, accurate, and economical manner.

(2) Multiplexer – To connect the data sources to the rest of the data acquisition system, there must always be some sort of multiplexer. In a conventional system using a general-purpose computer for the data acquisition task, the data sources are controlled and sometimes sampled in parallel, but the samples ultimately are brought into the computer one after another by sequential addressing of the data sources.

Analog multiplexers are in common use in the process control industry; a typical unit will, upon being presented with a 5-bit or 6-bit address, select one of 24 or 48 analog input lines, and connect the selected line to the output line with no significant loss

*Conversion of Gray code to standard binary code is a very simple operation; see for example pp. 229-234 and 399-401, Logical Design of Digital Computers, M. Phister, John Wiley and Sons, Inc., New York, 1958. Here, this conversion may be done in parallel for all Gray-code instruments.

in the accuracy of the signal and with a "settling time" delay of the order of a few microseconds. Digital multiplexers may be composed entirely of standard logic circuits, and can make a connection much faster than analog multiplexers since it is not necessary to allow a settling time as for a precise analog signal. Either type of multiplexer may be "random-access," as in the case of the typical "analog multiplexer" just described, in that any line can be addressed at random and will be connected and settled after a few microseconds.

Other multiplexers include within their own control logic the capability of automatically selecting lines in a fixed sequence. However, the sequencing logic or "programmer" which performs this task is a very simple device, and often provides only a mode of operation in which every line is selected exactly once during a given sampling sequence in some predetermined order.

In associative-computer-based systems, on the other hand, there is no hardware which can precisely be identified as the "multiplexer". The operation of "multiplexing" now comprises simply the readout of data, from the same field respectively, of each strand corresponding to a data source selected as a result of the search on whatever tolerance measure is used. In a sense, however, the associative computer functions as - among other things - a digital multiplexer.

(3) A/D Converter - In a system containing analog data sources, there must always be an analog-to-digital (A/D) converter. If the system is conventional and uses an analog multiplexer, ordinarily there is one high-speed A/D converter connected to the output of the multiplexer; the organization of the converter in this case is typically of the "successive-approximation" or "subranging" type (see

reference 7). If the system is conventional but uses a digital multiplexer, or if the system uses an associative computer or similar device as is indicated here, there must effectively be a separate A/D converter for each data source.

It is convenient in the case of the associative computer system to use a "counter" A/D converter organization (see again reference 7), and to let all of the associative computer strands share the counter and its associated digital-to-analog (D/A) converter. Thus, the only A/D converter circuit elements which must be duplicated for each strand are an analog comparator, and a flipflop (which is probably already part of the strand) to indicate when the analog value of the central counter equals that of the data source. A description of the applicable conversion process is given in Section III. E.

In any case, once the analog input has been converted to digital form, a "steering tag" must be associated with the digital value to identify which data source it came from. In a conventional system the steering tag is the address given to the multiplexer; in an associative-computer-based system it is a quantity stored permanently in one field of each strand and read out every time a data source is obtained from that strand, or defined in some other manner by the selection and detection unit (see Section III. D). Simplified data acquisition systems in which data sources are read out in a predetermined and inflexible order do not need to use steering tags, since the identity of the source which produced a given reading can be determined subsequently from the order of that reading in the sequence. However, such simplified systems cannot use any very powerful form of data compression; and the steering tag is after all only n bits for a system having up to 2^n data sources. Data-compression techniques would normally result in a large enough saving that this initial handicap, of extra bits for the steering tag, would not matter significantly.

(4) Sample-and-Hold Circuits - Certain problems of "time phase uncertainty" can arise when the sampled and transmitted data is reconstructed, and an attempt is made to plot the readings of many different data sources against time on one graph, because the sampling process itself may require an appreciable amount of time in relation to the rate in which the data itself is changing. To eliminate this problem, it is customary to provide "memory" at each analog data source for the analog signal, in the form of a sample-and-hold amplifier circuit.

Such a circuit may be thought of as a capacitor with a switch which may either connect it to the sensor, connect it to the analog multiplexer, or disconnect it from either to prevent it from discharging spontaneously; however, the actual form of the circuit is usually considerably more complicated. Thus, at some particular sampling time, all sample-and-hold amplifiers are connected to the sensors and allowed to stabilize, until their contents accurately represent the value of the sensors, and they are then all disconnected. The analog multiplexer can then sample the sample-and-hold amplifiers at its leisure, as long as a certain maximum permissible delay is not exceeded beyond which the stored signal may have decayed appreciably. The system logic assumes henceforth that the data in each sample-and-hold amplifier was obtained at exactly the same time.

A digital sample-and-hold register can be used for each binary sensor for a similar purpose. It is worth recalling here that the logic for conversion from Gray code to binary code is simple enough (see the most recent footnote) that this digital register could input in Gray code format from the data source and output in regular binary format to the multiplexer. In an associative computer, this "register" can of course, simply be one field of each strand, and Gray-to-binary conversion can be an internal associative computer operation.

(5) Buffer Memory - In a sophisticated data acquisition system, there must somewhere be a device which provides buffer memory to accrue the digitalized data as it becomes available, and which in turn gives commands to the multiplexer to select particular sensors at each given sampling time. In most larger conventional data acquisition systems this device is a general-purpose (GP) digital computer, much like any other such computer except for the much greater design emphasis which must be placed on real-time interrupt capability. In commercial data acquisition systems the final repository of the data within the system is normally a digital magnetic tape, compatible in format with the tape used by the large scientific computer which will subsequently be called on to analyze the data. In conventional systems using data compression, the data-compression function is performed either by a GP computer, or else by some simplified special-purpose computer; in either case, the data-compression computer is logically located after the multiplexer and the A/D converter, and before the magnetic tape unit. In the system of Section III of this report, the buffering function is shared by the associative computer and GP computer, and the final stage of the system aboard the vehicle is the transmitter; an output tape may be produced by the monitoring computer at the earthside tracking station.

An associative computer is well suited to perform several of the above functions at once. Each data source is permanently connected to one strand of the associative computer; a strand normally has its own reading, writing, comparison, arithmetic, and input/output capability on a bit-serial (sequentially, one-bit-at-a-time) basis. Assuming that the data is in digital form and available bit-serially, or else that the A/D conversion techniques of the next section are used, the associative computer can read in the data, perform a simple computation on it to obtain a normalized tolerance measure, search over the measures for all data sources in order to determine which readings must be transmitted during this sampling cycle according to the data-compression

criteria, and forward these readings to whatever device is next in line in the system - here, by assumption, a GP computer. The multiplexer and the data compressor have thus been merged into one device. The A/D converter can readily also be merged into it and thereby speeded up, as will be brought out in Subsection III-E.

The needs remain for a buffer memory and for a master control device; in future space vehicles, these needs are likely to be met by a GP computer, for reasons discussed in Subsection III-F, and also because of the currently prevailing "integrated avionics" philosophy of centralizing vehicle control functions in one relatively powerful computer. In order that the operation of the associative computer not interfere with that of the GP computer, the latter should have a "buffered" input/output control subsystem which operates independently of its arithmetic-and-control unit, and "steals" memory cycles whenever the associative computer needs to obtain an instruction or has data to be entered into the GP computer memory. The GP computer and the associative computer thus exist in a kind of "symbiotic" relationship.

The use of an associative computer in place of an ordinary multiplexer, besides merging several previously diverse system functions into one device, permits for the first time the effective application of data-compression criteria in which all of the data sources aboard the vehicle are considered jointly on each sampling cycle rather than each device being considered separately. Thus, the system has added capabilities not present in the "old style" system of Figure 2-1 in that, given sufficiently sophisticated data-compression criteria, there is now some guarantee that the use made at any given time of the data transmission channel for sending scientific data is "optimal" in the sense that always "the most important information is sent, and as much of it is sent as there is room for". Also, the problem of "buffer overflow" now vanishes, because the rate at which data is selected for transmission to ground can be made to depend on the rate at which it can be transmitted to ground. The price paid for eliminating buffer overflow is that a decrease in transmitter capability now leads automatically to the use

of looser tolerances, with some net information loss thereby resulting; however, there is every reason to prefer this condition to the uncontrolled and arbitrary information loss which results whenever buffer overflow occurs.

If the system is required to be able to continue acquiring data during a period of complete transmitter shutdown and to retain this data for subsequent transmission whenever the transmitter can again operate, or if it is intended to be able to avoid the use of very loose tolerances without the occurence of buffer overflow, there is an obvious need for a memory of larger capacity than the random-access memory of the GP computer or the specialized memory of the associative computer. Small, ruggedized digital magnetic tape recorders have been developed for aerospace applications, and one of these would normally be suitable. However, if a satisfactory solid-state mass memory with no moving parts could be developed to withstand an aerospace environment, it would of course be preferable.

The failure of one associative computer strand typically only takes out that strand and the corresponding data sources. If the associative computer has strand-to-strand shift capabilities (see Subsection III-D), it is possible - although not very rapidly executed - to read the datum obtained on each sampling cycle out of the failed strand into another strand which is operational, where the datum can be processed. Any more convenient form of strand redundancy would involve some sort of switching matrix, perhaps between strand memories and strand logics, which would introduce some undesirable complexity; very limited switching might perhaps be used for the strands corresponding to some of the most essential instruments. The central elements of the associative computer, such as the search, mask, and field strands, the A/D conversion counter, and the microstep decoding logic could be implemented using some form of logic-level or subsystem-level redundancy, such as majority-logic or quaddedlogic, if the application required strong measures to provide that degree of protection from failure.

SECTION III
SYSTEM DESIGN

A. SYSTEM ORGANIZATION

The Associative Data Acquisition System (Figure 3-1) consists of two major parts--an associative computer and a general-purpose (GP) computer. The associative computer provides the capability of monitoring and processing the inputs from many data sources, and the GP computer performs storage and control functions.

The specific functions of the GP computer include: program storage for both associative computer and GP computer instructions, storage for the acquired data, and over-all system control.. To perform these functions, data paths are provided from various points in the associative computer to two GP computer registers--the data register and the memory address register--as shown in Figure 3-1.

Access to the memory address register of the GP computer is required to allow the associative computer to obtain instructions from the random-access memory and also for the interchange of data between the two computers. The associative computer has its own sequence counter which shares access to the memory address register with the sequence counter of the GP computer. Thus the associative computer can sequence through its program by stealing memory cycles from the random-access memory.

The data register provides the path for instructions from the GP computer and for all data transfers. Instructions are sent directly to the associative computer control unit where they are used to initiate the proper sequence of microsteps. . Data transfers, however, are directly between the data register and the associative computer I/O register.

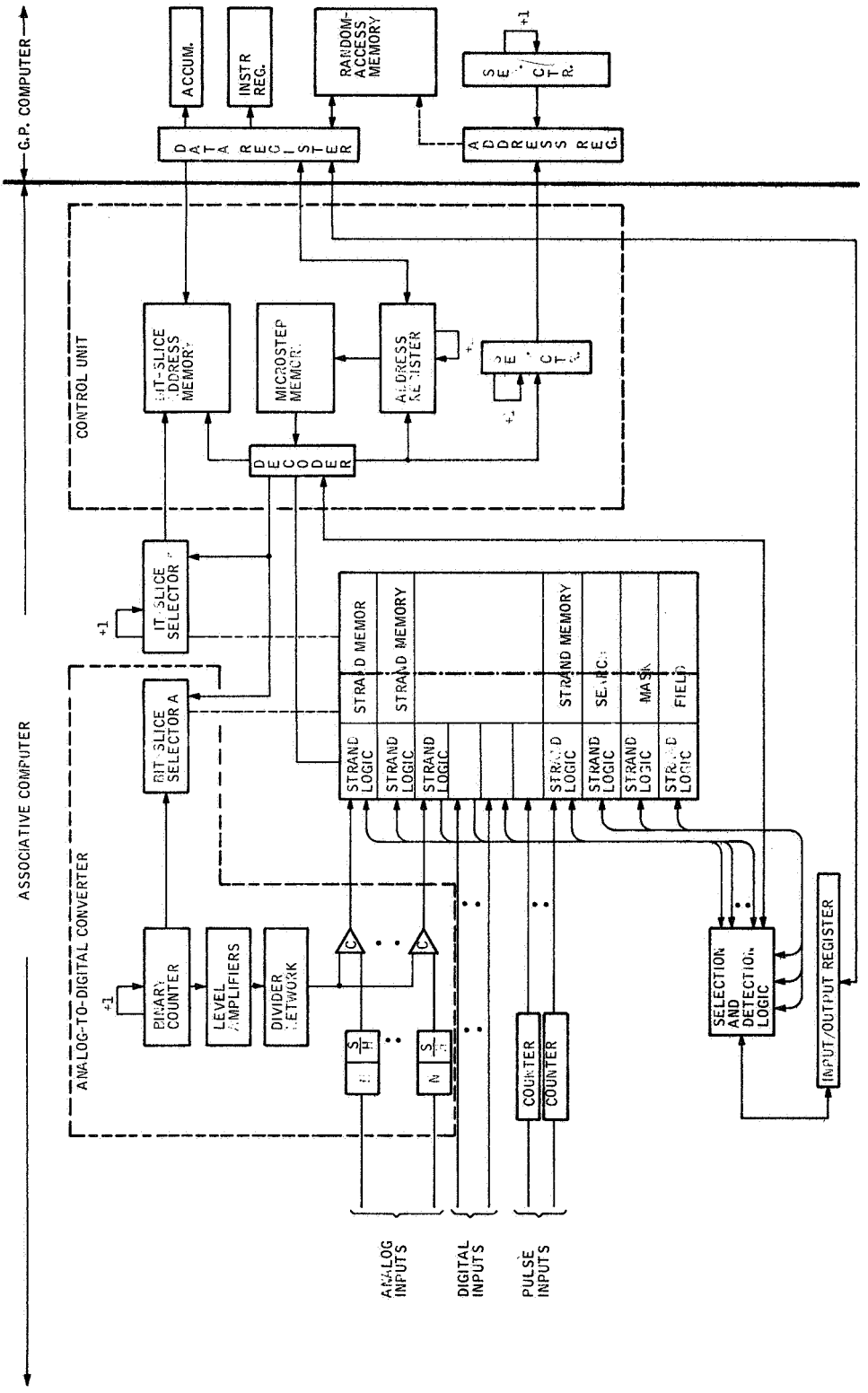


Figure 3-1. Data Acquisition System

The heart of the associative computer is an array of processing units called strands. Each strand is made up of one word of memory (normally several data fields), a block of logic which is capable of serial arithmetic as well as simple comparisons, and an input converter. These parts of the strand will be called strand memory, strand logic, and strand input converter, respectively.

To describe the reading and writing capabilities of the array of strands, the terms bit slice and word slice will be used. A word-slice read or write will involve many bits in the same strand (sometimes more than one strand for writing, if the same information is to be written in several strand memories at once); a bit-slice operation involves the same bit of all strands. The primary mode of operation for reading and writing as well as for processing is the bit-slice mode. The only requirement for word-slice writing during the analog-to-digital (A/D) conversion is described in detail in Subsection III-G. Bit-slice selector A is used to provide this word-slice writing capability. Bit slice-selector B is used to select the appropriate bit slice for all operations other than A/D conversion.

Three special strands--labeled the search strand, the mask strand, and the field strand in Figure 3-1--are required to allow the array of strands to perform the search operations of an associative memory. All or a portion of the contents of the search strand can be compared simultaneously to the contents of the other strands for equality, inequality, or any other search criterion. The mask strand determines the portion of the search strand to be used in the search operations. Flexibility in field length is provided with the field strand; a unit in a bit position signifies the beginning of a field and zeros are used as spacers. Arithmetic operations of two general types can be performed by each strand; these are called central and field operations. In a central operation each active strand furnishes one of the operands from its own strand memory, while the second operand is obtained from the search strand. In field operations each active strand obtains both operands from its own strand memory.

The inputs from data sources to the array of strands are in one of three forms: analog, digital, or pulse. The digital values are presented in a serial fashion and consequently are written a bit-slice at a time into specified portions of the digital strands. The random-pulse inputs are capable of extremely high burst rates, so counters are provided on each line to serve as "frequency dividers." The analog inputs are normalized, sampled, and temporarily "held" for comparison to a step voltage produced with a counter-type digital-to-analog conversion; a comparison match is followed by writing the value of the binary counter into the "matched" strands, using a word-slice write mode to write all bits of this value in parallel in each strand where a match occurred at the present count value.

The selection and detection unit performs several functions: It serves to detect any solutions in the strands in the course of a search operation, and the circuitry also allows read-out of the selected strand into the I/O register. It also provides for distribution of the search, mask, and field words to the strands for all central operations, and for distribution of the contents of the I/O register to the strands for writing purposes. Along with logic contained in each strand, it also provides for selection of strands one at a time for readout in a situation where more than one strand has been identified by a search operation as containing a solution.

The control unit of the associative computer is based on the use of micro-programming. The reasons for this approach are discussed in Subsection III-B. The microstep memory stores the various microstep (microinstruction) sequences necessary in the performance of all associative instructions. For each instruction received from the GP computer, the control unit sequences through an appropriate set of microsteps. When a given associative instruction is completed, the sequence counter is incremented and the next associative instruction is read out from the random-access memory. The one or more addresses included in the associative instruction are stored in the bit-slice address memory. During the execution of an instruction they are used by bit-slice selector B to select bit slices of the strand array. Each time an address is used it is incremented and returned to the address memory.

B. ASSOCIATIVE COMPUTER CONTROL UNIT

Control in the associative computer is accomplished with microprogramming. This approach allows flexibility in sequencing and also permits the use of subroutine structuring for most of the instructions implemented in the system. A more conventional hard-wired control unit might replace the microprogram system when the exact sequencing of all instructions has been operationally verified.

The main parts of the control unit are the "microstep memory" and the "bit-slice-address memory" as shown in Figure 3-1.

The microstep memory provides for storage of the sequences of microsteps required for each instruction executed by the associative computer. The value in the address register of the microstep memory locates a word which contains one or more microsteps. Each step is decoded to provide control in one or more subsections of the associative computer. The address register is incremented to obtain a sequence of microsteps for each instruction executed.

As can be noted in Figure 3-1, control is provided to the strand logic, to the bit-slice selectors, and to the selection and detection logic. The majority of these control or gating signals are used in the strand logic to enable the various search, arithmetic, reading, and writing operations to be performed. Gating signals are also provided to bit-slice selector A, to bit-slice selector B, and to the selection and detection logic. The exact number of signals and the function of each will depend on a number of design tradeoffs still being considered in these areas.

The microstep memory in the present configuration could be either a non-destructive-readout (NDRO) or a fixed ("fixed-information" or "read-only") memory. It might be desirable to select an NDRO memory in the first system design to allow alterations in the microprogram, and then to replace it with a fixed memory in later models.

The other major part of the control unit, the bit-slice-address memory, provides storage for the addresses contained in associative computer instructions. These addresses specify bit slices within the strand memory; as many as three addresses are required for some instructions. Initially, these addresses specify the least or most significant bit of each field involved in the instruction. They are incremented or decremented, depending on the operation being performed, each time they are used during the execution of the instruction. The address to be used during each microstep of a given instruction is specified by control signals from the microstep decoder. The mechanization and details of the design of this portion of the control unit have not yet been completed.

A list of instructions which the associative computer can be expected to be able to execute has been developed. While this list must be considered tentative at the present stage of the design, it is included to show the general capabilities of the associative computer. The instruction set is divided into six subsets. The first subset contains all arithmetic operations performed by the associative device; it is subdivided into two parts, one in which the operations are performed between fields in the strands, and the other in which the contents of the search register and the contents of one field are the operands. The second subset of instructions implements various writing operations which are needed. All search instructions are contained in the third grouping, and the subdivision is like that of the first subset. The fourth subset is composed of all instructions affecting the contents of the enable flipflop (EFF). All logic instructions involving the field segments of the strands are found in the fifth subset, and again a further division exists to separate the "field" instructions from the "central" instructions involving a segment of the search word and a field. All special instructions, such as the command to sample the data sources, are to be found in the final subset. In all listed instructions, the strand participation is dependent upon the value of the enable flipflop (EFF); if an unconditional operation is desired, a "U" is prefixed to the mnemonic, which causes the machine-language associative computer instruction after assembly to have a "one" instead of a "zero" in a particular bit of the operation code field.

Table 3-1 lists some of the associative computer instructions which might be implemented. This list is not claimed to be complete, nor is it implied that all of the listed instructions will be included in the final design.

Under the "Effect" heading in Table 3-1, the letters A, A', B, C, and C' denote fields in strand memory and S represents the search strand; parentheses around a letter designate the contents of that field. A' is the next field after A, and likewise for C' and C. The arrow in each operation statement represents the expression "and the result is stored in".

"E" denotes the value of the EFF in the strand logic, and control bits in the search memory are designated as "V". "T" stands for the TFF in strand logic.

In the "Timing" column, "n" signifies the length of a single field participating in the operation; "R" and "W" respectively denote the time required for a bit-slice read operation and a bit-slice write operation in the memory.

C. ORGANIZATION OF STRANDS

Each strand of the associative computer is divided into three parts: (1) strand memory, (2) strand logic, and (3) strand input converter.

1. Strand Memory

The memory portion of each strand consists of 192 memory elements, one for each bit slice of the array. All memory elements in the strand memory must be randomly accessible for reading and writing operations.

Table 3-1. Instruction Set

Group	Op Code	Name	Effect	Timing
I Arithmetic Operations	FAD	Field Add	$(A) + (B) \rightarrow C$	$3nR + nW$
	FDA	Field Double Add	$(A) + 2(B) \rightarrow C$	$3nR + nW$
	FSU	Field Subtract	$(A) - (B) \rightarrow C$	$3nR + nW$
	FDS	Field Double Subtract	$(A) - 2(B) \rightarrow C$	$3nR + nW$
	FMU	Field Multiply	$(A) \cdot (B) \rightarrow CC'$	$(n^2+n)R + \left\lfloor\frac{n^2+n}{2}\right\rfloor W$
	FDV	Field Divide	$(AA') \div (B) \rightarrow C$	$2n^2R + n^2W$
	FSR	Field Square Root	$\sqrt{(AA')} \rightarrow B$	$2n^2R + n^2W$
	FAO	Field Add One	$(A) + 1 \rightarrow A$	$nR + nW$
	FSO	Field Subtract One	$(A) - 1 \rightarrow A$	$nR + nW$
	FAV	Field Average	$\frac{1}{2}[(A) + (B)] \rightarrow C$	$3nR + nW$
	CAD	Central Add	$(A) + (S) \rightarrow B$	$2nR + nW$
	CSU	Central Subtract	$(A) - (S) \rightarrow B$	$2nR + nW$
	CIS	Central Inverse Subtract	$(S) - (A) \rightarrow B$	$2nR + nW$
	CMU	Central Multiply	$(A) \cdot (S) \rightarrow B$	$\left\lfloor\frac{n^2+n}{2}\right\rfloor R + \left\lfloor\frac{n^2+n}{2}\right\rfloor W$
	CDV	Central Divide	$(A) \div (S) \rightarrow B$	$n^2R + n^2W$
II Write Operations	FCL	Field Clear	$0 \rightarrow A$	nW
	FMV	Field Move	$(A) \rightarrow B$	$nR + nW$
	FMT	Field Move Twice	$(A) \rightarrow B, C$	$nR + 2nW$
	CWR	Central Write	$(S) \rightarrow A$	nW
III Search Operations	FGT	Field Greater Than	$1 \rightarrow T$ if $(A) > (B)$	$2nR$
	FGE	Field Greater Than or Equal	$1 \rightarrow T$ if $(A) \geq (B)$	$2nR$
	FLT	Field Less Than	$1 \rightarrow T$ if $(A) < (B)$	$2nR$
	FLE	Field Less Than or Equal	$1 \rightarrow T$ if $(A) \leq (B)$	$2nR$
	FZE	Field Zero	$1 \rightarrow T$ if $(A) = 0$	$(n-1)R$
	FNZ	Field Non-zero	$1 \rightarrow T$ if $(A) \neq 0$	$(n-1)R$
	FPS	Field Positive	$1 \rightarrow T$ if $(A) \geq 0$	R
	FNG	Field Negative	$1 \rightarrow T$ if $(A) \leq 0$	R
	FEQ	Field Equality	$1 \rightarrow T$ if $(A) = (B)$	$2nR$
	FIQ	Field Inequality	$1 \rightarrow T$ if $(A) \neq (B)$	$2nR$
	CGT	Central Greater Than	$1 \rightarrow T$ if $(A) > (S)$	nR
	CGE	Central Greater Than or Equal	$1 \rightarrow T$ if $(A) \geq (S)$	nR
	CLT	Central Less Than	$1 \rightarrow T$ if $(A) < (S)$	nR
	CLE	Central Less Than or Equal	$1 \rightarrow T$ if $(A) \leq (S)$	nR
	CEQ	Central Equality	$1 \rightarrow T$ if $(A) = (S)$	nR
	CIQ	Central Inequality	$1 \rightarrow T$ if $(A) \neq (S)$	nR
	CMA	Central Maximum	$1 \rightarrow T$ if (A) Max.	nR
	CMI	Central Minimum	$1 \rightarrow T$ if (A) Min.	nR
IV Enable Operations	ESS	Enable Set	$1 \rightarrow E$	-
	ERR	Enable Reset	$0 \rightarrow E$	-
	ECM	Enable Complement	$\bar{E} \rightarrow E$	-
	ESA	Enable Save	$(E) \rightarrow V$	W
	ERC	Enable Recall	$(V) \rightarrow E$	R
	EAN	Enable Logical AND	$(E) \cap (V) \rightarrow E$	R
	EOR	Enable Logical OR	$(E) \cup (V) \rightarrow E$	R
V Logic Operations	FAN	Field Logical AND	$A_i \cap B_i \rightarrow C_i$	$2nR + nW$
	FOR	Field Logical OR	$A_i \cup B_i \rightarrow C_i$	$2nR + nW$
	FER	Field Exclusive OR	$(A_i \cap \bar{B}_i) \cup (\bar{A}_i \cap B_i) \rightarrow C_i$	$2nR + nW$
	FCO	Field Coincidence	$(A_i \cap B_i) \cup (\bar{A}_i \cap \bar{B}_i) \rightarrow C_i$	$2nR + nW$
	CAN	Central Logical AND	$A_i \cap S_i \rightarrow B_i$	$nR + nW$
	COR	Central Logical OR	$A_i \cap S_i \rightarrow B_i$	$nR + nW$
	CER	Central Exclusive OR	$(A_i \cap \bar{S}_i) \cup (\bar{A}_i \cap S_i) \rightarrow B_i$	$nR + nW$
	CCO	Central Coincidence	$(A_i \cap S_i) \cup (\bar{A}_i \cap \bar{S}_i) \rightarrow B_i$	$nR + nW$
VI Miscellaneous Operations	SDS	Sample Data Sources	-	-
	FRD	Field Round	$(A) + 1 \rightarrow A$ if $A'_{n-1} = 1$	$(n+1)R + nW$
	GBC	Gray-to-Binary Conversion	-	-

A data transfer path exists from each element of a strand to the strand logic. There is no hardware requirement for element-to-element data transfers within a strand. Transfers between strands need be implemented only if they are desired either for reliability or other reasons, or because of the choice of the "shift" approach to selection and detection (see Subsection III-D). In any case, transfers between strands would involve strand logic and not strand memory.

Considered collectively the strand memories of a 256-strand associative computer consist of a 192 x 256 array of memory elements. Random access to a bit slice (one of 192) is necessary for data transfers between a single memory element of each strand and its logic, simultaneously for all strands. The enable or interrogate signal to the selected bit slice is furnished by bit-slice-selector B (Figure 3-1).

In addition to the requirements described above, writing in a parallel-by-bit (word-slice) mode is necessary in one particular field of the strand memory during the analog-to-digital conversion process. In this case the information to be written is furnished from bit-slice-selector A, rather than from the strand logic. Whether the elements in this portion of the strand memory are identical to the other elements depends on the choice of memory components and on the memory circuit design.

The bit-at-a-time operating mode of the associative computer places certain requirements on the memory element; it must have fast read and write times compared with conventional memory elements, it must consume relatively little power, and it should be non-volatile. For these reasons, plated wire appears to be the leading contender for the memory component at present.

2. Strand Logic

The strand logic contains the circuits necessary to perform the required read, write, search, and arithmetic operations. The main information flow paths in this part of a strand are illustrated in Figure 3-2. While this basic configuration

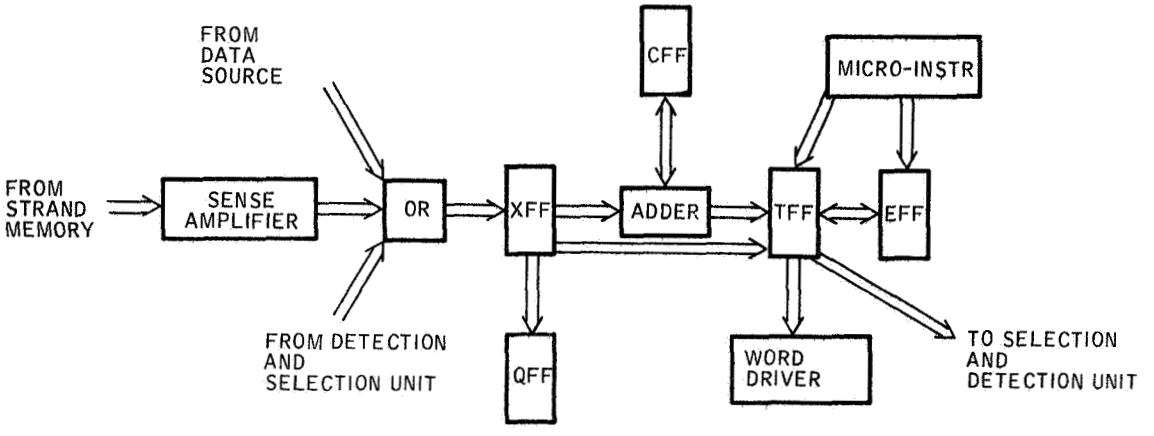


Figure 3-2. Information Flow Diagram for Strand Logic

may be slightly altered after more detailed examination, this structure seems at the present time to offer distinct advantages.

The associative computer operates in a "bit-slice" mode, so that the information flow begins at the readout of the particular bit or at the readout of a bit in the search strand; the value of the bit is first transferred to the exchange flipflop (XFF) to effect a storage of the bit during a bit-slice sequence. From this location the bit value can be used in an addition operation; it can also be transferred to the quotient flipflop (QFF) during multiplication or division sequences, or transferred to the tag flipflop (TFF). The carry flipflop (CFF) stores the partial carry during the "half add" operation and also the carry into the next most significant bit position at the conclusion of each bit operation. The TFF is also loaded with intermediate and final results of microstep sequences, so that the content of the TFF must sometimes be written into strand memory and must be tested at some other times during various associative computer instructions. The enable flipflop (EFF) determines whether that particular strand is to be active during a phase of the instruction sequence. It is loaded from the TFF, or controlled by special microsteps, during instruction execution.

The strand logic for the search, mask, and field strands does not require arithmetic capability, and so a different configuration might be implemented in these strands. The only parts of the strand logic necessary in these strands would be a sense amplifier, an exchange flipflop (XFF), a tag flipflop (TFF), and a word driver. The contents of a bit location would be stored in XFF for subsequent use by the selection and detection logic. The TFF would allow write-in of data through its connection to the word driver. Since the XFF content in the search and mask strands must be widely distributed, logic amplifiers may be needed to provide fanout. Special circuits for the search, mask, and field strands might be developed, or the standard strand logic might be used with alterations in the control connections; a more detailed investigation must be made before reaching a decision on this matter.

3. Strand Input Converter

There are three types of data inputs to the associative computer: analog, digital, and pulse. Both the analog and pulse inputs require "conversion" before they can be submitted to strand memory. This conversion circuitry is another part of the data strand and is called the strand input converter.

The strand input converter for analog inputs will consist of the necessary normalizing, sampling, holding, and comparator circuits for the type of analog-to-digital conversion implemented in the system; a detailed explanation of three promising conversion schemes is the subject of Subsection III-E, "Analog-to-Digital Conversion", and of Appendix C, "Pulse-Type Analog-to-Digital Conversion".

The strand input converter for straight digital inputs could in some cases be eliminated entirely; in other cases, a set of level-changing logic amplifiers and/or a set of flipflops for holding might be required. If certain digital inputs were only available in Gray code, the necessary Gray-to-binary conversion might be performed as an associative computer algorithm.

Since the burst rate expected in many of the pulse inputs will exceed the sampling rate of the associative computer, the strand input converter for pulse inputs must function as a "frequency divider". Counters will accumulate the number of pulses received between associative computer sampling cycle and allow entry of this count into a strand memory at the request of the associative computer.

D. SELECTION AND DETECTION UNIT

The selection logic enables all information transfers to occur within the associative computer and between the associative computer and the other devices of the system; and the detection logic allows sensing of the results

of searches. The two are intermixed, so that it is useful to class them for description purposes as the "selection and detection unit". The unit transfers information from the input/output register to particular strand memories, and conversely. Distribution of the search and mask bits to all strands is effected by the selection and detection circuitry, and detection of field terminations is performed and transferred to the control unit for subsequent action. The included logic must be able to detect the presence of one or more solutions in strand logic and notify the control unit accordingly. The various strands satisfying a search operation must be located and read into the input/output register.

Three different approaches to implementation of the selection and detection logic have been investigated, and the merits of all three must be considered before a final selection can be made. These alternative approaches, categorized by their strand selection technique, are the scan approach, the shift approach, and the propagation approach.

1. Scan Approach

The scan approach provides selection and detection with an "OR" tree which has as its inputs identical-per-strand logic outputs; the basic circuitry is shown in Figure 3-3. Each "SUBSET" line permits an "OR" logic operation to occur over the particular subset of TFF's.

Assume now that a search operation has just been performed, which has left a "one" in the TFF of each strand whose memory contents satisfied the scan criteria. A strand whose memory contents have satisfied a search operation may be selected as follows: The "ALL" line is energized, thus gating all TFF outputs into the "OR" gate. If any "one" is present, the detect flipflop will be set; the search then sequentially activates the "SUBSET" lines until a "one" is again detected. A counter is set to the

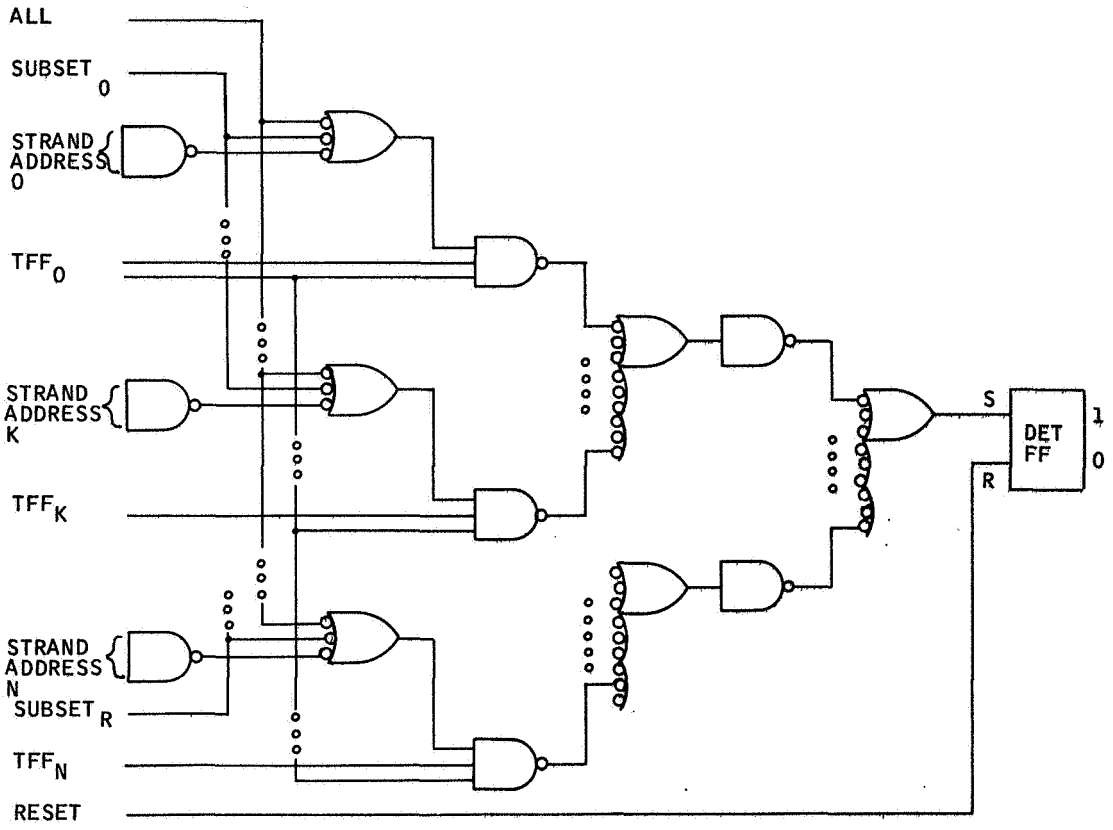


Figure 3-3. Scan Approach – Selection and Detection Logic

beginning strand address in the subset of strands where the first solution has been found, and the strands are individually interrogated until the first "one" in a TFF of that subset is detected in order to locate a strand whose memory contents satisfied the search; the readout of this strand can then be performed bit-serially through the same "OR" tree. To write information into an individual strand, data is transferred bit-serially from the input/output register to a strand determined by the value of the counter. For specific searches and "central" arithmetic operations, the search and mask bits are gated to all data strands from a logic amplifier in the selection and detection unit; the presence of a "one" in the field strand indicates to the decoding unit that the end of the specified field has been reached. Some advantages of this particular approach are: the modularity of the design; the one-clock-time detection of any solution to an associative sequence; and the explicit random-access readout and write-in provided.

2. Shift Approach

The shift approach would be a natural one where for some other reason the associative computer is required to be able to shift information between different strands. A shift register is formed using the TFF from each strand, as indicated in Figures 3-4 and 3-5. Each TFF can shift its content to its neighbor in either the up or the down direction, and certain other shift capabilities are also provided. Specifically, for a 256-strand associative computer, every fourth TFF would be able to shift to the TFF 4 positions away in either direction, every sixteenth TFF to the TFF 16 positions away in either direction, and every sixty-fourth TFF to the TFF 64 positions away in either direction. If the associative computer were intended to do certain tasks unrelated to data acquisition which required it to function as a square array (for example, matrix inversion), probably every TFF and not just every sixteenth TFF would be able to shift its contents 16 positions up or down; in the general case for a 2^{2n}-strand square-array associative computer, the shift would be every 2^nth TFF to the TFF 2^n positions up or down.

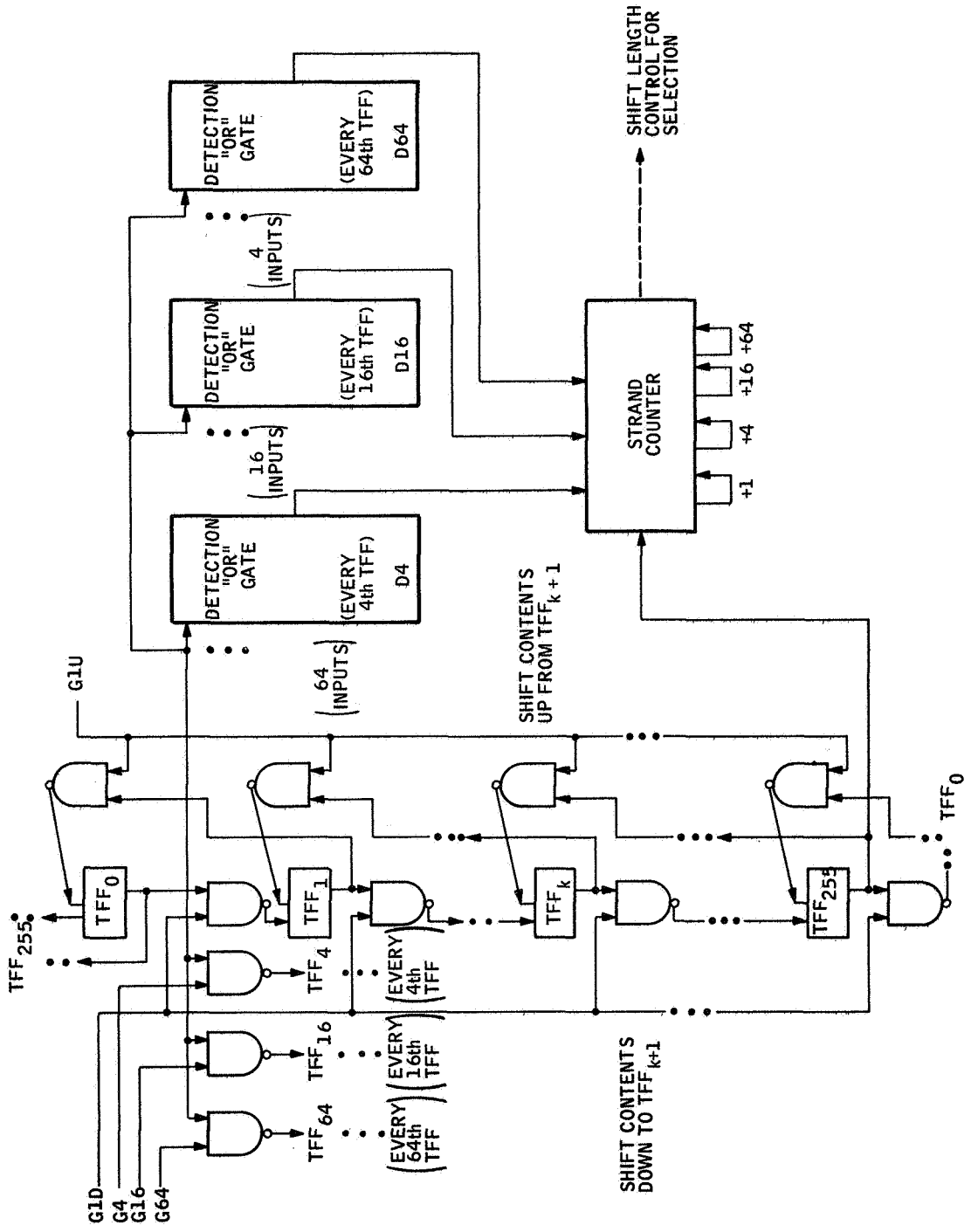


Figure 3-4. Scan Approach – Selection and Detection Unit

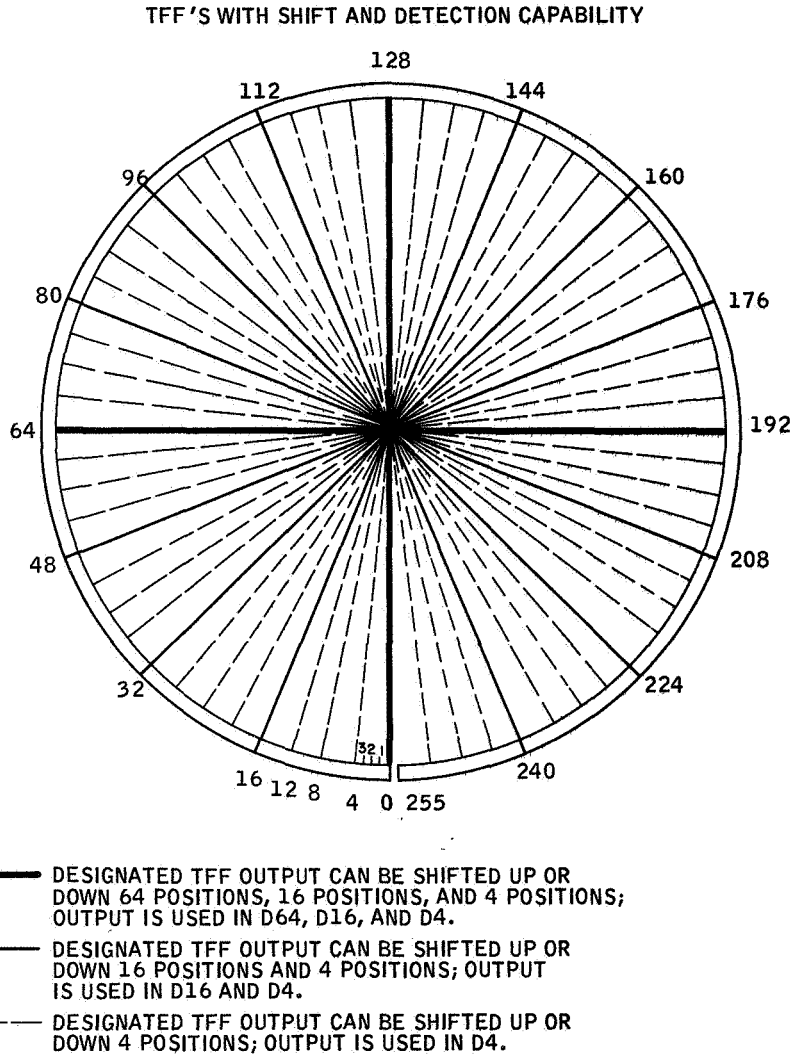


Figure 3-5. Shift Interconnection Diagram for Shift Approach

It is assumed in the following description of operations that the information in the TFF of each strand has been duplicated either in another flipflop or in a memory bit of that strand, so that the information in the shift register does not need to be preserved after a detection or selection operation has been performed. Also, every fourth TFF is connected to a detection "or" gate D4, which here has 64 inputs; every sixteenth TFF is connected to another "or" gate D16 having 16 inputs; and every sixty-fourth TFF is connected to a 4-input "or" gate D64.

Detection of the existence of at least one "one" in some TFF is performed as follows: The contents of each TFF are shifted into its neighbor in both directions (up and down), without clearing any TFF, for two consecutive bit times. After this has been accomplished, any "one" which existed prior to the shift must have entered one of the TFF's connected to D4, and the output of D4 thus indicates the presence of a "one" in the shift register although the contents of the shift register are now "hash" (i.e., are not meaningful). This technique reduces the time to detect a "one" in the shift register to two bit times, which is important since maximum and minimum search operations require such a detection operation for every bit of operand. This bidirectional, "non-erasing" shift operation can readily be performed with shift-register flipflops which obey the following truth table;

"L-C" Flipflop Truth Table

Load Input L	Clear Input C	Content, n^{th} Bit Time $Q^{(n)}$	Content, $n + 1^{th}$ Bit Time $Q^{(N + 1)}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The first seven lines of the truth table are identical with that for a J-K flipflop; the last line is different, since if both inputs of a J-K flipflop are true the content Q of the flipflop is complemented, whereas for an "L-C" flipflop Q remains the same. The "L-C" truth table is believed by many in the computer industry to be the most desirable one for a flipflop which is to serve as an element of a parallel data register or a shift register. Its major advantage is that shifting from one flipflop to the next, or from one parallel register to the next, can normally be done simply by turning on all clear inputs and connecting the true output of one stage to the load input of the next; with the clear inputs turned off, information is retained as in an R-S or J-K flipflop. With R-S or J-K flipflops, two couplings per stage of shift are needed rather than one -- the true output of one stage to the S or J input of the next, and the false output of one stage to the R or K input of the next. A minor amount of external logic on a J-K flipflop enables it to conform to the L-C truth table; in the case of the Sylvania SUHL II J-K flipflop which has been used as a representative microcircuit unit in our design studies, most of the required logic is already or the flatpack.

Selection of the strand corresponding to the first "one" in the shift register may be performed as follows: All TFF's are shifted down one bit at a time until a "one" is detected by D4 or until 4 places of shift have occurred without any such "one" being detected (in which case there are no "ones"). Then every fourth TFF is shifted down four bits at a time until a "one" is detected by D16, and the process is repeated for 16-place and 64-place shifts until a "one" is detected by D64 and in the end position respectively. A counter keeps track of how many places of shift were necessary to bring the given "one" to the end position. This counter would here need to count in increments of 1, 4, 16, and 64, which would not be difficult if it performed counting by propagating a carry through half-adders instead of using "count logic," since it would now suffice to be able to introduce a carry at any of four binary positions.

The simplest, and slowest, practical technique for input to or output from a strand would be to shift one bit of information at a time, decoding the shift count into 2-bit groups and using each group to control the corresponding shift operation of the proper length. If the logic is made sufficiently sophisticated to shift always in the direction of the nearest point at which a longer shift can be performed for output, and to use a similar minimal shift strategy for input, a bit can be shifted all the way to or from the end position in at most eight shift times. With no extra control but some caution in the choice of strategy, up to four bits in a group may be shifted in 11 shift times by this same technique. However, unless some selective means of clearing the shift register at certain TFF's is provided, there cannot be more than four bits in such a group or the unintentional "or-ing" of information will occur. Shifting a 12-bit number would, for instance, then require either shifting three 4-bit groups, or the use of some other strategy - such as only using the 1-place and 16-place shifts, which would allow a group of up to 16 bits to be handled in at most 32 shift times.

This scheme is the most conservative of the three, since it uses synchronous logic with low fan-ins and fan-outs throughout except for the "or" gates - and these gates are smaller than the 256-input gate in the previous scheme. The scheme could be implemented with very little hardware in an associative computer already endowed with the shift capabilities necessary for matrix inversion. Although it is the slowest of the three schemes, it would not be a great deal slower if enough different-length shift capabilities were included and the logic clock rate were appreciably higher than the memory read rate.

3. Propagation Approach

The propagation approach to the selection and detection logic is based on the use of iterative circuit techniques. One of the many forms of the logic for an iterative circuit cell that will perform this task is shown in Figure 3-6.

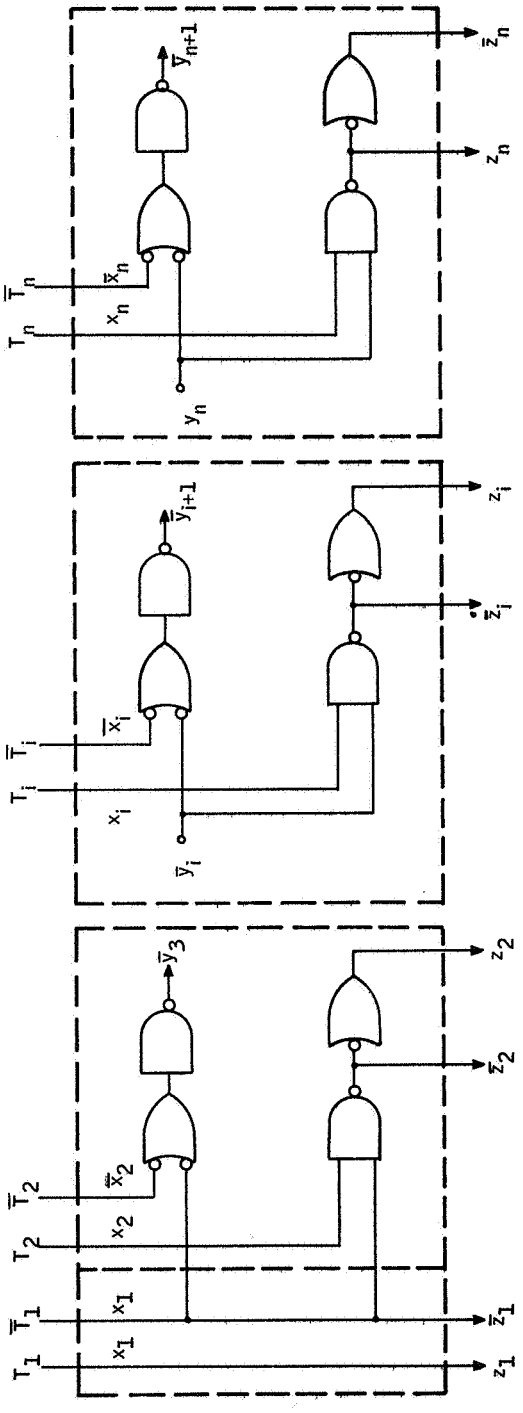


Figure 3-6. Logic Diagram of Iterative Circuit for Selection and Detection

Each cell has two inputs (a carry from the previous cell and input from the TFF) and two outputs (a carry output to the next cell and a parallel cell output). The Y_i carries are "zero" until the first TFF holding a "one" is reached, after which all of the remaining carries are "ones". Therefore if the last carry is a "one" it signifies that at least one of the TFF's contains a "one". The parallel cell output, Z_i, is a "one" only if it is obtained from the unique cell whose associated TFF contains the first "one" in the T register. The term "T" register is used here to denote the collection of TFF's, and the search for "ones" in this "register" may arbitrarily proceed from either end of the "register".

The mechanization illustrated in Figure 3-6 would probably be satisfactory only for small memories, since the carry propagation delay for circuits employing high-speed TTL would be about 10 to 12 nanoseconds per strand, A memory with 100 strands would then require 1 microsecond to perform the selection and detection while a 1000-strand memory would require about 10 microseconds.

This delay can be reduced considerably with only a very slight increase in complexity using carry-speedup techniques. One method of achieving this is illustrated in block-diagram form in Figure 3-7 and is given in logic-diagram form as part of Figure B2 in Appendix B. A series-parallel carry-generation scheme is used with the series network generating the carry for every seventh stage, thus reducing the propagation delay by a factor of seven. The number of logic elements is no greater than that required by the completely serial (iterative) network, but the maximum fan-in (the number of inputs to one of the elements) is increased from 2 to 8. A further reduction in delay could be attained by using expander circuits to increase the number of inputs to this element. With a fan-in of 8, the delay for a 1000-strand memory would be less than 2 microseconds; both the detection of "ones" and the selection of the first strand containing a "one" can be accomplished in this amount of time.

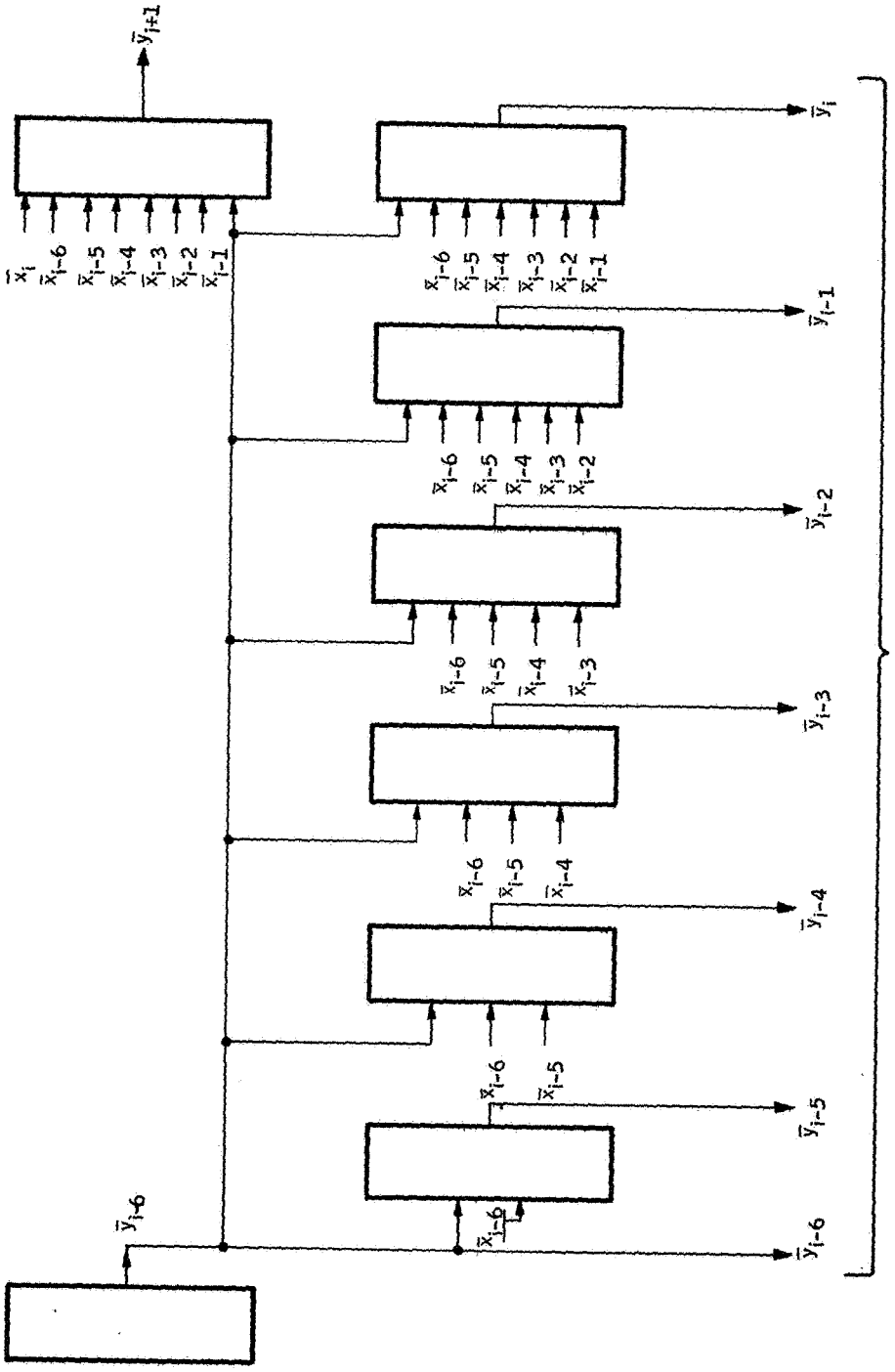


Figure 3-7. Block Diagram of Logic with Carry Speedup

E. ANALOG-TO-DIGITAL CONVERSION

Some of the data sources which will be used aboard space vehicles are inherently digital; some present their reading as a mechanical displacement or shaft rotation, so that their electrical output may be either analog or digital as desired: and some are intrinsically analog devices. The problem is then to connect each type of data source to the associative computer in the most natural, economical, accurate, and efficient way.

All analog data source outputs must first be converted to digital form. Either (1) each analog data source must have its own analog-to-digital (A/D) converter; (2) an analog multiplexer must be connected to all analog data sources in such a way that a single central "successive-approximation" or "subranging" type (see Ref. 7) A/D converter can perform all required conversions, following which the converted values must be distributed ("demultiplexed") to the proper associative computer strands; or (3) some scheme must be used whereby each strand has its own A/D conversion capability, but a great deal of the required hardware is shared among all strands.

Alternative (1) is appealing from a reliability viewpoint, but probably too expensive in terms of the size, weight, and power consumption of the resulting equipment. Alternative (2) is very unappealing from a reliability viewpoint, since the failure of the centralized A/D converter would make all analog instruments henceforth useless on any unmanned mission where in-flight repair is impossible; at the very least, the centralized part of the system would have to be duplicated. Moreover, alternative (2) is grossly awkward from a system design viewpoint, since the operation of the entire associative-computer-based data acquisition system presupposes that data sources and strands are matched one-for-one, and hence the multiplexing and demultiplexing stages represent "waste motion". Alternative (3) is thus the logical choice; the particular approach taken requires relatively little per-strand hardware, and the central elements shared by all strands can readily be duplicated or triplicated for reliability.

The essential principle of partially decentralized A/D conversion is to generate a central reference voltage signal, whose digital equivalent is continuously available, and distribute it to all strands where A/D conversion is required. Each strand then has its own comparator, to match the central reference against the voltage signal received from its corresponding data source. The equipment used to produce the central reference voltage can be duplicated or triplicated, and bridge circuits can be used to match the duplicate or triplicate outputs. In the case of triplication, if one voltage differs sharply from the other two the average of the latter is probably still correct, and hence a kind of "majority voting" is applicable.

The central reference voltage signal is changed according to some "strategy" to minimize conversion time and maximize accuracy. Two strategies will now be described: the "simple counter," and the "sectioned counter". If it were true that only certain analog values were ever assumed, some other strategy could be preferable; Reference 7 is a good guide to the strategies in use today.

It will be assumed that an accuracy of 1 part in 10^3 (or about 1 part in 2^{10}) is required, so that a 10-bit binary counter suffices. This counter comprises 10 flipflops plus associated count logic, and its value may thus range from $0 = 0000000000$ to $2^{10} - 1 = 1111111111$. Since flipflop outputs are not nearly precise enough to serve as accurate analog signals to be summed for D/A conversion of the counter value, it is necessary to use a level amplifier with each flipflop; this type of circuit connects in a reference voltage (here assumed to be 10 volts for descriptive convenience) wherever the flipflop contains a one, and connects in ground whenever the flipflop contains a zero. The outputs of all level amplifiers are connected to a ladder network (see Figure 3-8), whose output then is the exact analog equivalent of the binary number in the counter register; it can be shown that the most significant level amplifier is weighted 1/2, the next one is weighted 1/4, the next one 1/8, and so on down to $2^{-10} = 1/1024$ for the least significant one. Thus, for instance, if the counter contained $1011000000 = 11/16$ of full scale, the ladder network output would be $(1 \times 5.000) + (0 \times 2.500) + (1 \times 1.250) + (1 \times 0.625) + 0 + \ldots + 0$ or 6.875 volts which is 11/16 of the full-scale 10-volt reading.

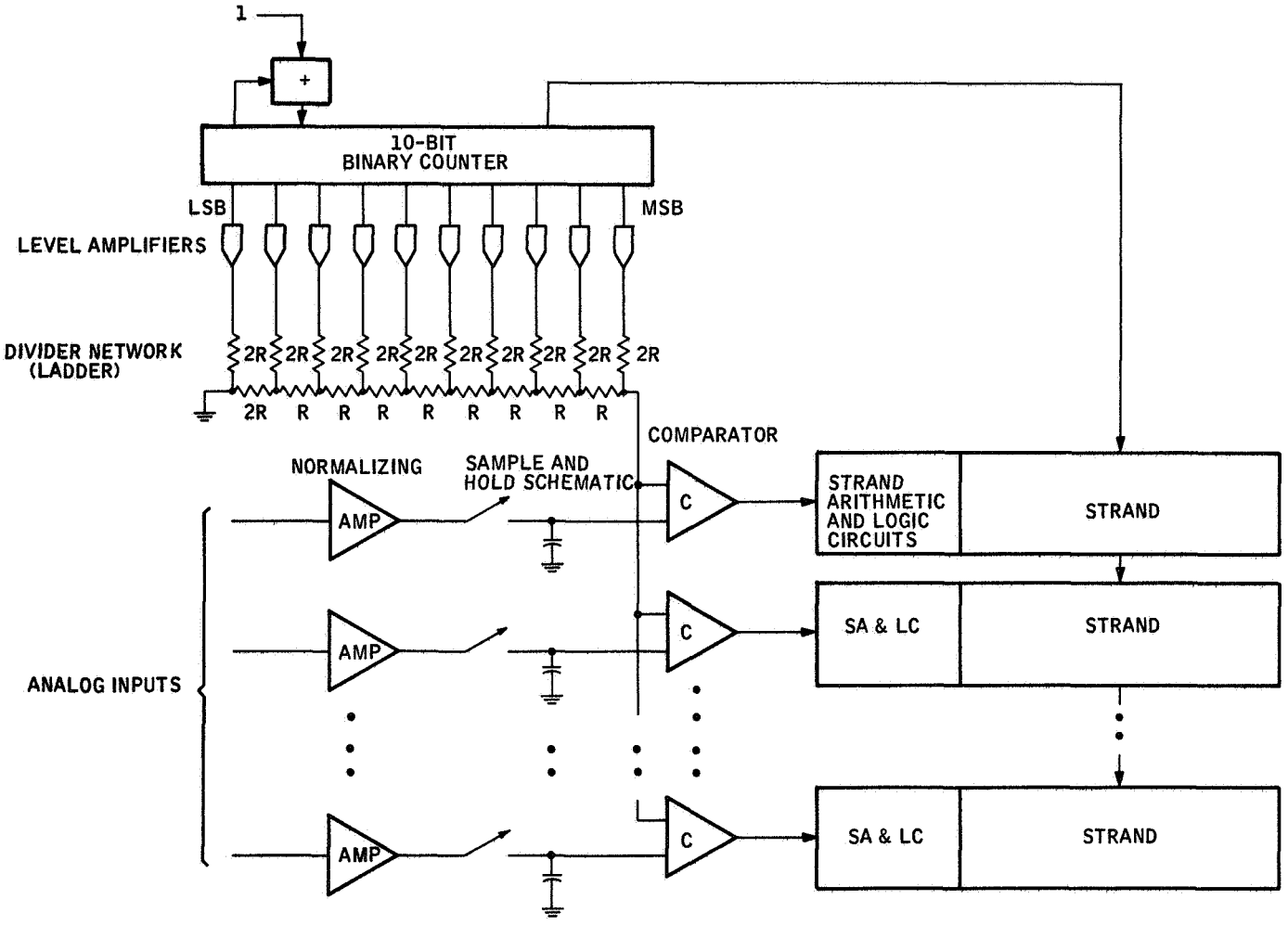


Figure 3-8. Counter-Type A/D Converter

The "simple counter" scheme for converting analog inputs to digital form is illustrated in Figure 3-8. The associative computer has a sufficient number of strands so that each analog input can be connected to its own analog-input strand, and each digital or pulse input can be connected to its own digital-input or pulse-input strand. (Besides these types of strands, there are still the three special strands - search, mask, and field - which are used for implementing associative memory operation: the search strand stores any word to be compared to all others in the memory, the mask strand stores a word which specifies that certain bit positions (corresponding to zeroes in the mask word) are to be ignored in a search, and the field strand defines the boundaries of individual pieces of data in all of the other strands..

Each analog input is first isolated and normalized to a standard voltage range (such as 0-5 volts) by a normalizing amplifier. At some specified time, all inputs are sampled and stored in their respective sample-and-hold amplifiers. Conversion may then be accomplished basically as follows: A binary counter counts from 0 to $2^n - 1$ where conversion is to be done to a precision of n bits. The value of this counter is continuously converted to analog form by a digital-to-analog (D/A) converter, and this analog form is distributed to all analog-input strands. The counter counts at a uniform rate; depending on the speed of the analog circuits, its analog output has a "ramp" or a "staircase" form. As the counter counts up in increments of 0000000001, the ladder network output is thus a very accurate analog equivalent of the counter value; its behavior with time is that of a "ramp" voltage, possibly with stair-steps if the response of the A/D converter (the level amplifiers plus the ladder network) is fast enough.

Each strand includes a two-input analog comparator, which produces a logic output of zero or one according to which of its inputs is higher. Such a comparator can now be made as a single integrated circuit. The central reference voltage is connected to one comparator input, and the voltage output from the data source corresponding to that strand is connected to the other one.

It is now apparent why all data source analog outputs should be "normalized" to a specific range, determined by the characteristics of the central reference voltage; if the latter ranges from 0 to almost 10 volts, so should the former. An added benefit of normalization is that each data source output is converted to binary with the maximum available precision, which would not be the case if for instance some data source outputs ranged only from 0 to 2 volts and maximum precision corresponded to 10 volts; for this reason, such normalizing amplifiers are also desirable in conventional data acquisition systems.

As the ramp voltage assumes every value in its entire range, every per-strand comparator will eventually change state. When this change occurs for a given strand, the comparator logic output changes, and this change sets a flipflop which in turn controls the per-strand driver circuit; a "crosswise write" operation (see Appendix B of Ref. 1) then takes place which writes the value of the counter into a data field of that strand (and of any other strands whose comparators likewise just changed state - see below.). The flipflop is then reset, and its input logic is such that it cannot be set again until the comparator output has again changed state twice - once when the central reference voltage drops to zero at the end of the current conversion cycle, and again when the comparison is first satisfied during the following conversion cycle. One advantage of this scheme is that, given adequate per-bit ("interrogate") drivers, if several analog inputs are equal so that the comparators of several strands change state at the same binary counter value, this value can be written into a data field in every one of these strands simultaneously using the crosswise write technique. Performing a crosswise write operation to write the complete counter values requires twice as long as performing a normal "bit-slice" write operation; however, an entire data field is written, rather than just one bit of many data fields in parallel. It may be seen that, when the binary counter reaches full scale, all analog data sources will have been converted to their binary equivalents with the latter stored in the same data field of the corresponding associative computer strands.

It is desirable to use only one time tag for all data source readings obtained on one A/D conversion cycle; when a few of these readings are selected for transmission, only one copy of the time tag need be sent with all of them, thus achieving a non-trivial saving in the number of bits which must be transmitted to ground. In order to avoid uncertainty in the exact time at which a reading was taken ("time-phase uncertainty"), it is desirable that all data sources be sampled simultaneously. Hence each data source should be equipped with a sample-and-hold amplifier, which can store its output accurately and can continuously furnish this output to the corresponding per-strand comparator for the duration of one A/D conversion cycle.

It should be observed that in the simple counter scheme, the sample-and-hold amplifier for the data source reading is only required for synchronization purposes; the time when a reading was taken is very accurately determined in any case. If a sample-and-hold amplifier were not used, the time when each reading was taken could accurately be related to the time origin at the beginning of the staircase, since for the simple counter scheme the time when conversion occurs is directly proportional to the value converted.

There are two "sectioned counter" approaches; one of these is a direct modification of the "simple counter" approach. As the scheme will be described here, the counter is divided into two sections (upper and lower), of five bits each, which count independently. The upper section counts first, and digital storage of only the upper five bits of the counter is performed when a comparison occurs on a strand. The analog counter output is stored in a second sample-and-hold amplifier in that strand, and the contents of the first sample-and-hold amplifier are subtracted from the contents of the second one to produce a "difference" voltage. This voltage is then converted to digital form on a second pass, in which the lower section alone counts. The final converted value for each strand is then obtained by a "field subtract" operation, in which the less significant converted result is subtracted from the more significant one.

A modification of this approach, featuring pulse techniques, reduces the number of sample-and-hold amplifiers required to one per analog-input strand but raises additional accuracy problems in the analog portion. This approach was developed by W. A. Plice of the Honeywell Aeronautical Division on Honeywell funds, and is described in Appendix C.

If the central reference counter were to count once every two microseconds, which is reasonable with contemporary components, a complete conversion of all analog inputs to 10-bit accuracy would require 128 microseconds using either sectioned counter scheme (assuming a 5-5 partition, and 2048 microseconds using the simple counter scheme). A reduction of the conversion accuracy requirement to eight bits would reduce these times to 64 microseconds (assuming a 4-4 partition scheme) and 512 microseconds respectively. Assuming that there are 128 analog inputs, the net conversion time per input (as a basis for comparing the associative memory approach with a conventional approach where the conversions are sequential) would be as follows:

	Simple Counter	Sectioned Counter
8-bit Accuracy	4 microseconds	1/2 microsecond
10-bit Accuracy	16 microseconds	1 microsecond

F. GENERAL-PURPOSE COMPUTER CHARACTERISTICS

It has been assumed that a general-purpose computer exists in the data acquisition system, along with the associative computer. Since the characteristics of suitable general-purpose computers are well understood, this GP computer has properly not been the object of much study in itself. It is nevertheless useful to state here what sort of device this GP computer is presumed to be, in a system intended for use during the time period 1975-1985. A much

smaller-scale device, referred to as a "buffer memory" but serving much the same function, is described in Reference 4; since this buffer memory would need to be in service well before the end of the current decade, the ground rules in the system described in Reference 4 are very different from those here.

The GP computer would probably have a word length somewhat greater than the maximum anticipated length of a data field in a given strand, and further constrained by the requirements of error-detection-and-correction codes; the word length including check code might range from 24 to 32 bits depending on the application. It would use a random-access memory composed of elements suitable for non-destructive-readout (NDRO) operation, but also capable of fast writing. The most obvious such memory element at the present time is a plated-wire (cylindrical thin film) array; other possibilities include flat thin films, ferroelectric devices, electroptical devices, very small and highly developed multiaperture cores, and laminated ferrite sheets. Depending on what else the computer was required to do, the random-access portion of its memory might contain from 4,096 to 32,768 words.

It is possible to use a destructive-readout (DRO) core memory in the GP computer if this memory is equipped with a "program restore feature" such as is described in Reference 4. The essential idea of this feature is that a special drive wire threads all cores of words-to-be-protected, in such a way that a drive pulse on that wire will automatically write "ones" and "zeroes" into those cores in the proper pattern to reproduce the program. Although readout is destructive, in that the program information is destroyed after being read and must be regenerated as in the non-protected portions of the memory, it is obviously possible to periodically re-ensure the correctness of the program information by activating the program restore driver. By 1975, of course, NDRO memory components will no doubt have improved to the point where the use of DRO core memories in space vehicle digital computers memories will not be attractive; however, DRO core memories currently appear to have the smallest physical size, weight, and power consumption requirements, and the best resistance to extreme environments, of commercially available aerospace digital memories.

The GP computer would be capable of parallel or semi-parallel fixed-point arithmetic so that its rate of performing addition, subtraction, multiplication, division, square root, and logic and control operations would be relatively high; but it would probably be capable of performing floating-point operations only on an interpretively programmed basis. To facilitate the use of an especially convenient form of interpretive programming, it is important that those operation codes (in instructions) which are not recognized by the hard-wired machine logic be used as addresses for the most commonly used subroutines. This type of feature is in common use in real-time computers as of this writing, under such trade names as "programmed operator", "programmed instruction", "extracode instruction", "stored-logic instruction", and "pseudoinstruction".

It is very important that the memory of the GP computer be equipped with input/output capabilities which function autonomously with respect to the arithmetic-and-control unit. One of these channels, used entirely for outputs, leads to the transmitter. Another, which must be bidirectional, enables communication with the mass memory, which in earlier systems would probably be a magnetic tape unit and in later systems would perhaps be an electro-optic mass memory. Another bidirectional channel communicates with the associative computer, transmitting commands and receiving data samples.

The associative computer also has independent access to the GP random-access computer memory by a different and completely independent channel, controlled by its own "sequence control register" or "program counter." From the viewpoint of the GP computer, the associative computer gets instructions from the random-access memory by a process much like a normal output process, but there is one important difference; it is possible for the associative computer program to jump, which means to load a new value into the program counter so that the output process stops at the address of the jump instruction and starts up again at a different specified address. Even conditional jumps are possible on this basis, except that the conditions cannot be of the usual arithmetic sort encountered in general-purpose computers, since such conditions now occur independently in each strand of the associative computer.

One other important feature of the general-purpose computer is the ability to detect and act on unscheduled program interrupts. These may be of several different kinds: the programmed-instruction interrupts already referred to; external interrupts from other devices such as the transmitter, command receiver, or mass memory, which indicate that a status word should be requested from that device by the general-purpose computer; time interrupts received from the associative computer, which is more completely described in Section IV-B; interrupts from the autonomous input/output controllers, indicating that a data flow process has been completed or that a check-code error has occurred; and internally generated interrupts indicating some fault condition within the GP computer itself such as an attempt to divide by zero or to take the square root of a negative number, a divide overflow, a check code error, and so forth.

All data entering or leaving the GP computer memory passes through a "central data bus" or "exchange register", whether the information is going to or from the GP computer instruction register, the associative computer instruction register, the GP computer accumulator, or an input/output channel. The use of check-code features is highly desirable in memories, since even an occasional "false alarm" indication of an error when none exists is much preferable to the occurrence of undetected errors. Contemporary ground-based computers use parity-checking extensively, but rarely use any more ambitious form of coding. A suitable error-detection-and-correction encoder for a smaller-scale computer in a similar system is discussed in Reference 4. To indicate what sort of code-checking measures are now available, the three examples given in Reference 4 will be quoted here; however, the information-part length of 10 or 11 bits is not adequate for the GP computer in this system.

Code	Number of Check Bits Required	Number of Information Bits Possible	Correction Capabilities	Detection Capabilities	Burst-Error Correction Capabilities
a. Hamming binary code of distance 4	5	11	single error	double error	2 bits or less
b. Hamming cyclic code of distance 4	5	10	single error	double error	5 bits or less
c. Bose-Chaudhuri code	4	11	single error	triple error	4 bits or less

It is anticipated that efficient operation of the GP computer will require that certain special addresses not be stored within the memory proper, but instead be more rapidly accessible in a separate "address storage module" device. Special addresses include the current program sequence addresses for the GP and the associative computers, the input/output channel "current" and "terminal" addresses (see below), and possibly others. A small random-access integrated-circuit scratchpad memory would probably be a suitable address storage module. This module would need to be equipped with a full-address-length binary counter, capable of counting up an address used for reading an instruction, or for an input/output memory access, before the address was replaced in storage.

The simplest way to control an autonomous input/output channel is the following: a "current address register" contains the address of the next data word to be accessed, and a "terminal address register" contains the address of the last word of the block to be input or output. Whenever the current address equals the terminal address, the input/output process is terminated, and an interrupt is generated for detection by the GP computer

arithmetic-and-control unit. A somewhat more sophisticated input/output control would be desirable here; in particular, the capability should exist for counting the current address backwards, in order that data can be recorded on a magnetic tape traveling backwards (at least, if magnetic tape storage is to be used aboard the space vehicle).

The functions performed by the general-purpose computer are diverse. It is ultimately responsible for the formatting of all information selected by the data acquisition system for transmission. It reads data from the associative computer by means of search operations at the end of each sampling cycle, attaches a time tag obtained from the central space vehicle time counter to these data, and keeps with the data "steering tags" or "strand-address numbers" which identify each datum as belonging to some given data source.

Also, there may be some types of data whose basic form is changed by the GP computer prior to transmission. One such task for the GP computer is the conversion of certain pieces of data from fixed-point to floating-point form, where a decrease in the number of bits to be transmitted can thereby be achieved, consistent with acceptable rendering of the data; there are, for instance, certain types of charged-particle counts which may vary over 4 or 5 decimal orders of magnitude, but for which the experimenters desire only a small fixed number (often 6 or 7) of significant bits.

The GP computer initiates input/output processes involving its own memory, but does not have to supervise these processes while they are going on. At the conclusion of one, however, the computer is interrupted, and must reset the current address and possibly also change the terminal address if the next input/output process is to be from a different part of the memory.

The GP computer is also responsible for obtaining those commands from the command receiver which are intended to affect the operation of the data acquisition system, and seeing to it that these commands get carried out. Whenever the command receiver receives a relevant command, it should generate an interrupt; the command may then be read by the general-purpose computer as a status word.

The possibility of using this GP computer for other space vehicle computation and control functions, besides those relating to the data acquisition system, has not been considered on this project. It is well worth noting, however, that many other tasks such as guidance and navigation which must be performed aboard the space vehicle could also be carried out by this same GP computer. One task particularly appropriate for this computer is the execution of system self-test programs to verify that the GP computer, the associative computer, and other subsystems are operating properly and (for analog devices) are still in correct calibration.

SECTION IV
SYSTEM OPERATION

A. ACQUISITION CYCLE

The over-all cycle of the data acquisition and compression system is comprised of the following steps: the measurement step, the input step, data conversion, data compression, and the output step.

(1) Measurement Step -- To ensure a proper time relationship between the data values of all sensors, the measurements for the various inputs will occur at the same instant and will be "held" at that value until being submitted to the strand array.

(2) Input Step -- There is such a marked difference, between the sampling sequence for the analog data sources and the sampling sequence for the digital and pulse data sources, that these operations cannot occur simultaneously. After the measurement step has been completed, the measured values for pulse and digital inputs will be written bit-serially into those strand memories which are enabled. As this sequence is taking place, the analog values are being held; the analog-to-digital conversion is then performed in the next part of this step in accordance with the methods described in an earlier section of this report, and the data is entered into the strand memories requiring input.

(3) Data Conversion -- After all required strand memories have been loaded, a conversion step consisting of one or several conversion instructions is effected to put the contents of the strand memories into proper form. One such conversion instruction might convert Gray-coded (reflected-binary) data to binary.

(4) Compression -- The next step in the system applies data compression algorithms to the strand memory contents; some of these algorithms are found in Appendix A of Reference 3. The results of the data compression are a "figure of merit" and the data which is to be transmitted; both values are stored in fields of the strand memory.

(5) Output Step -- The "figures of merit" for all data strands are examined with a search operation to determine the data sources which are to be sampled. This method of selecting data points to be transmitted, based on an examination of all figures of merit simultaneously, is sometimes referred to as the use of "all-sources-jointly" techniques; further details may be obtained from Reference 3.

B. SPECIAL DATA SOURCES

1. Raster-Type Instruments

"Raster" instruments such as television cameras, certain types of radar apparatus, and scanning radiometers pose special problems in a data acquisition system because of the vast amount of data involved. While methods of handling this type of instrument will be considered more extensively during the remainder of the project, some initial thoughts on the subject are described below.

The feature of the associative computer that can be utilized for the pictures that result from the use of these instruments is its highly parallel structure. An 800 x 800-point TV picture, with six bits required to represent the "gray level" at each point, might be considered almost typical. Since the rate at which points are being produced by the TV camera is considerably higher than a complete cycle of the associative processor, real-time acquisition and processing necessitates the use of many strands of associative computer. One way of handling this would be to provide one strand for each column of points on the picture; therefore, 800 strands would be required for the 800 x 800-point

picture. Since the points are produced by the camera sequentially along a given raster (horizontal line), the 800 points resulting from one scan along the raster line must be distributed to the 800 strands of associative computer. Each point will be stored in the analog sample-and-hold amplifier which is a part of the input converter of each strand. A second set of sample-and-hold amplifiers must be provided to store the points produced by the next raster line while input conversion, compression, and output selection is being performed for the first set. Thus, the functions of the two-per-strand sample-and-hold amplifiers will alternate from one cycle to the next.

Assuming a computational cycle of 500 microseconds for the associative computer (the exact time depends upon the method of analog-to-digital conversion used and on the data compression algorithm chosen), an 800 x 800-point video frame or still picture could be processed in 800 x 0.5 = 400 milliseconds, for a rate of 2-1/2 frames per second. This rate is insufficient for flicker-free real-time TV, but by a small enough factor that the use of additional parallelism could suffice to achieve an adequate frame rate.

2. On-Demand Operation of Instruments

There are certain important experiments for space-vehicle exploration missions which do not require continuous monitoring of instruments by a computer, since the instruments either are not functioning continuously or do not have meaningful information for output at all times. For example, in certain life-detection experiments samples are acquired and placed in a special apparatus whose outputs are then examined at some later interval, possibly 20 minutes later. Another important example of a discontinuous experiment is active seismological exploration of a planet; a seismic transducer probe is installed in the ground, explosive charges are distributed at various distances from the probe location, and as these are set off the output of the transducer is monitored. During the period when the effects of the explosions are still being felt, the amount of significant data to be acquired and processed is very great; but at other times,

when the seismic probe is functioning in a purely passive experiment, the amount of significant data acquired might be so small as to be entirely negligible if the planet was undergoing no important volcanic activity or crust shifting. Another example occurs on a planet "flyby" mission; at a certain point in the trajectory it may be possible to observe the sun through the edge of the planetary atmosphere, and then to determine in what way this atmosphere selectively absorbs portions of the solar output spectrum.

To properly sequence such experiments, a powerful and flexible timing device is needed. The associative computer is capable of functioning as such a timing device, although the way in which it is used for this purpose is radically different from the way it gathers samples from continuously-readable analog or digital data sources. A space vehicle or roving vehicle must normally have its own central frequency standard or time counter, unless it can communicate with such a standard aboard another vehicle close by; in any case, the vehicle must at any instant be in possession of very exact time information, usually expressed in binary form. The binary time count may easily be as long as 35 to 40 bits, since it is necessary to keep track of time periods of the order of months or years down to a resolution of the order of one millisecond.

Whenever it is desired to initiate some experiment at some future time, the current time count is read, an increment is added to it to specify a suitable future time, and this future time "alarm setting" is stored in a strand memory. On each sampling cycle, or at some other convenient interval, all future time alarm settings are compared with the current time count using an "equality search" operation. Whenever a match occurs, an interrupt is generated and sent to the GP computer, as has already been mentioned in subsection III-F.

The requirements of this task are radically different from those of the normal data handling function; for instance, they presuppose a different field length. It remains possible to perform these timing functions within the same associative computer; however, there must now be two field strands, one to define

the fields for the strands used for normal data acquisition and compression computations, and the other to define those for future time alarm settings.

The same strand used to generate the future time alarm for a particular instrument or experimental apparatus might, of course, also be used to receive data from that instrument or apparatus; in fact, it might even be used to transmit commands to the apparatus. However, it would be equally possible to use this timing technique to control instruments not able to communicate with the associative computer at all. Some rarely used instruments might be connected directly to the GP computer on a "roll-call" basis, which means that many instruments are connected to one data channel, but each instrument is activated only when its particular address is presented on the GP computer's "external function" lines by a GP computer instruction.

C. ACQUIRED DATA HANDLING PROCEDURES

While one of the significant features of this data acquisition system is that it automatically matches the amount of data acquired to the current transmission capability, it is assumed that situations may still occur where some data must be stored for transmission at some future time. For instance, the transmitter may fail temporarily because of radiofrequency interference, or because the vehicle is passing through a planet shadow and its solar cells are not producing enough power to enable transmission. Therefore, a bulk storage or "mass memory" device such as a magnetic tape unit is assumed to be in use.

The three possible modes of operation that are required are described below:

(1) Acquiring Rate Equals Transmission Rate -- The most common situation is where the data-acquiring rate of the associative computer and the transmission rate are equal.

(2) Acquiring Rate Exceeds Transmission Rate -- In this case, more data is being gathered than can be handled by the transmitter. A complete

communications blackout, for instance, will require that all gathered data be stored for later transmission. It is also possible that even though the transmitter is operable, the control procedures implemented in the data acquisition system still determine that some storing of data in the mass memory is preferable to attempting to "hard-limit" the amount of data acquired. For instance, if matching the acquiring rate to the transmission rate requires corridor widths in the compression algorithms that exceed the "recommended" corridor width to the extent that excessive "filtering" is being applied to the data, then the system may elect to automatically shift to the "acquiring rate exceeds the transmission rate" mode and must then store some of the acquired data for later transmission. The most "significant" data values acquired during each sampling cycle will still be transmitted; the others will be accumulated in a buffer area of the random-access memory of the GP computer. When the buffer area is filled, it will in turn be emptied into the mass memory.

(3) Transmission Rate Exceeds Acquiring Rate -- In this case, some of the data previously stored in the mass memory can share the transmission channel with the data currently being acquired. The data acquisition system will now operate as closely as possible to the "corridor width limits" mentioned previously, the buffer area will be used to read from the mass memory, and the GP computer will include as many data points from the buffer as is possible for each transmission after a sampling cycle. Each time the buffer becomes empty, a new block is transferred from the mass memory to the buffer area, until all stored data points have been transmitted. The system then reverts to mode (1), either by reducing the corridor width tolerances or by reducing the transmission rate.

SECTION V
CONCLUSIONS AND RECOMMENDATIONS

This section contains a summary of the space vehicle instrumentation studies, which have been completed, and of the data acquisition system design, which is partially completed.

It also includes a description of the remaining tasks of the present program, which comprise portions of the system design plus a computer simulation of the data compression procedures carried out by the system. Lastly, some recommendations are made for future work.

A. SPACE VEHICLE INSTRUMENTATION

A classification scheme which has stood up well as our studies progressed is the following:

(A) High-speed or variable-speed continuously readable data sources

(B) Low-speed continuously readable data sources

(C) Data sources which can only be read after a certain delay following their receipt of a command

(D) Data sources corresponding to experiments which are only done at very infrequent intervals

(E) Anomalous data sources which present a non-numeric interface

Data sources of classes (A) and (B) would be tied into the associative computer in the "normal" way described in Part III of this report. Those of classes (C) and (D) must operate "on demand", and would be tied in as described in Section IV.B of this report. Those of class (E) probably could not profitably be tied in via the associative computer at all, but should be connected directly to the GP computer.

It appears fairly certain that the maximum precision required of analog spacecraft instruments is in the neighborhood of 12 bits, which is a little less than the maximum precision customarily sought in ground-based systems. Ten bits is probably enough for the vast majority of readings, and eight bits is enough for many. However, there are certain pieces of information, such as the time count and the reckoned space vehicle position, which must be known to much greater accuracy; and there are also random pulse counts and frequency counts which may range up to 20 bits or so, even though not all of these are considered significant enough to merit transmission back to earth.

The design of space vehicle instruments should be influenced by the idea that these instruments will be under the control of a digital computer; some saving in ad hoc digital logic within various instrument packages may thereby come about. Also, instrument designers should be prepared to state certain characteristics for their instruments which must be known in order to successfully operate them in a computer-controlled mode: precision (in equivalent binary bits), maximum interval during which the instrument may be left unsampled (that is, minimum sampling frequency), mode of operation (continuous, continuous at certain times, intermittent, occasional, one-shot), delay (if any) between the receipt by the instrument of a command to supply a reading and the availability of the reading, relevant special characteristics (for instance, pulse

counter readings always increase and never decrease, unless of course overflow occurs), and any simplifications or changes which are desirable in the instrument whenever it is to be used in a computer-controlled system.

B. CHARACTERISTICS OF THE PROPOSED SYSTEM

The fundamental organization of the associative computer has been defined, and many details of the over-all system design and operation have been determined. The plated-wire memory element offers distinct advantages over other NDRO devices at the present time, and hence it is the likely choice for use in the strand (associative computer) memory. Microprogramming allows control sequence flexibility, logic design simplicity, and ready alterability of the instruction set; thus control of the associative computer will reside in the microstep control unit. The required steps in the system operation of the associative computer have been defined, thus forming a base for coding instruction sequences necessary to operate the data acquisition and compression system.

Basic work on the system organization has produced various approaches to several subsystems of the associative computer; further investigation is therefore necessary to arrive at the best configuration for these subsystems. In particular, several methods of implementation have been proposed for the selection and detection logic, the analog-to-digital conversion subsystem and the strand logic.

Other parts of the data acquisition and compression system have not yet been considered in detail. The strand input converter for pulse data sources must be designed such that no pulses are lost in the acquisition cycle. The best data compression algorithm must be determined, and must then be programmed using the final associative computer instruction set. The microstep sequences for all instructions must be formulated after the microinstruction list has been completed.

C. SIMULATION INVESTIGATIONS

In the proposed Associative Data Acquisition System (ADAS), the data compression algorithms described in Appendix A of Reference 3 are used in a manner slightly different from the conventional way. This modified use of the algorithms is different in just one respect, but the difference is essential; in the ADAS approach, the results obtained by applying the algorithm to all data sources are considered relative to one another or jointly, rather than treating each result independently as in the conventional approach.

The advantage provided by the modified algorithm is really a kind of "look-ahead" feature; that is, the system can conveniently evaluate the figures of merit obtained from the compression computations not only to determine which data sources must be sampled during the current cycle, but also to predict how many might require sampling on future cycles. This predicted information can be used to smooth out the "peaks" in terms of the number of data sources whose output is chosen for transmission after each sampling cycle, either by altering the width of corridors used in the compression algorithm, or else by making a decision to either store information in the system mass memory or to recall information already stored. The over-all objective of the proposed simulation is to evaluate the usefulness of this look-ahead future. This evaluation will be done by comparing the ADAS approach and the conventional approach, using "experimenter tapes" (magnetic) of actual data gathered by satellites.

The phases of simulation effort are described below in outline form. Since several data-compression algorithms are to be considered, the procedure will be repeated for each selected algorithm.

(1) Program the conventional and ADAS versions of the selected algorithm, determining a method of assigning initial corridor values for each data source.

(2) Evaluate procedures for adjusting the corridor width in the ADAS version. The look-ahead feature will be used to initiate corridor width changes. The method of selecting those data sources on which changes are to be made will be determined.

(3) Evaluate the ADAS version by comparison with the conventional algorithm.

  (a) Display results of both and observe differences

  (b) Calculate and compare RMS errors between original and compressed data for both algorithms.

(4) Evaluate the use of special procedures; for instance: Implementing the rejection of "wild points"; limiting the length of runs from each data source; and precompression filtering.

D. RECOMMENDATIONS FOR FURTHER WORK

It is expected that by the completion of Phase II of this contract a good understanding will have been gained of the possible role of an associative computer in a space-vehicle instrumentation system. A number of possible follow-on programs are suggested here for NASA's consideration for future work on Phase III:

(1) Detailed prefabrication design

(2) Application in biomedical instrumentation systems

(3) Construction and checkout of a complete system

(4) Design studies of using associative computers for data compression in real-time remote TV systems

(5) Hardware development for high-speed distributed A/D conversion

(1) Detailed prefabrication design -- The type of design and simulation being done during Phase II will not result in a complete design which could be immediately fabricated. The logic design being performed is mostly at the system level and only partly at the logic-element level; and the simulation being performed is of the over-all functions performed by the associative computer and the execution of algorithms, rather than the flow of control and information signals through the logic.

Hence it is proposed that the present design efforts be carried further during another phase, to the obvious next step of performing a detailed logic design and a bit-time-by-bit-time logic simulation. This simulation would not be at the level of executing algorithms, but strictly at the level of executing associative computer microsteps; its purpose would be that of ensuring the correctness of the logic equations defining the design. Concurrent with this effort, microprograms for all common associative computer instructions would be coded; these would also be checked out by simulation. Both the bit-time-by-bit-time logic simulation and the checkout of microprograms would be carried out using a digital computer.

This task is prerequisite to performing task (3).

(2) Applications in biomedical instrumentation systems -- The instruments which have been investigated during Phase II of this contract have all been spaceborne instruments, such as might be used on an exploratory orbiter, lander, or deep space probe, with some minor attention being given to surface exploration vehicles. Another area where the associative computer approach to data compression looks as if it might be fruitful is that of biomedical instrumentation systems.

Hence, the survey could reasonably be extended to the instruments customarily used in biomedical work at such facilities as the Honeywell

Systems and Research Division Manned Systems Sciences Section, the University of Minnesota Medical School, the Mayo Clinic, and the NASA Manned Space Flight Center in Houston. This study would be greatly aided by the ready availability of consulting help from Honeywell biomedical researchers and experimental psychologists of the Manned Systems Sciences Section of the Research Department.

Task (2) might also, under the circumstances indicated below, be a prerequisite to task (3).

(3) Construction and checkout of a complete system -- Given the performance of task (1), and possibly that of task (2) if the intended application were in a biomedical instrumentation system, our technology has reached the point that construction and checkout of a realistic and operationally useful associative computer can then follow. This associative computer would be developed complete with the required analog-to-digital conversion apparatus; and Honeywell would, if desired, furnish a computer suitable for tie-in as the cooperating general-purpose computer.

To avoid the necessity for confronting the special problems of severe-environment component choice and packaging at this stage, the solution of which would considerably increase the cost of the prototype, it is recommended that the application for this prototype system be chosen from the biomedical field, since such a choice would usually permit the system to be used on the ground in a relatively favorable environment. On the other hand, if it were desired, a space-vehicle data processing system could instead be chosen as the prototype application and a proper aerospace packaging job could be performed.

In the latter event, task (3) would only depend on the prior completion of task (1) and not on that of task (2).

(4) Design studies of using associative computers for data compression in real-time remote TV systems -- A very important problem now on the horizon is the transmission of real-time TV information back to earth from space vehicles exploring other celestial bodies. If performed in the most straightforward way without the use of data compression, this type of video transmission would involve tremendous data rates, possibly as high as a billion bits a second. Such a rate is intolerable for the foreseeable future, and obviously some form of data compression will have to be employed to make such transmission feasible.

One possible technical approach is to use an associative computer in the manner which has been studied during Phase II of this contract, and in effect assign one "column" of a TV picture composed of raster lines in the "row" direction to each individual strand. For instance, if there were 1024 raster lines in the Y direction, and an equal resolution were desired in the X direction, the associative computer would have 1024 strands. This approach can provide compression of still pictures or individual video frames.

A number of variations of the above approach can be used to obtain compression between one picture or frame and the next. The data compression algorithms in this case can be applied to corresponding data points on successive pictures rather than along columns of points within a picture. This approach, while more difficult in terms of hardware requirements, would be particularily effective in the situation where the TV camera is in a fixed position, since large portions of the picture could be expected to remain constant for relatively long periods. For instance, a sequence of frames might show a fixed landscape with an astronaut walking across it, and only the portion of the picture showing the astronaut would have to be sent for each successive frame. Since it would be unreasonable to provide an associative computer strand for each point of the

picture, some method of time-sharing of strands would have to be used. In situations where the camera is moving, these same techniques might still be used, by noting that the picture has been translated horizontally or vertically by a certain number of resolution-matrix points between one frame and the next, so that the repeated portions of the earlier frame can be repositioned and reused.

Various combinations of within-a-frame and frame-to-frame compression techniques could certainly provide an enormous degree of video data compression. It is recommended that these techniques be studied, and the form of the required associative computer be investigated.

(5) Hardware development for high-speed distributed A/D conversion -- Two high-speed associative computer analog-to-digital conversion schemes, based on the use of a "sectioned counter", have been formulated recently at Honeywell. More, or else more complex, circuits are required to implement these schemes than are required for the "simple counter" scheme which has received most of our attention; but the sectioned-counter schemes result in very great speed gains as is discussed in Subsection III.E.

If task (3) were felt at this stage to represent too large a program, the analog-to-digital portion of it could reasonably be broken out as a separate smaller study project. It is not, of course, certain that the system proposed in task (3) would necessarily have to use sectioned-counter conversion techniques; the simple-counter technique might be sufficiently fast for a biomedical application.

One of the sectioned-counter schemes is described in Subsection III.E. of this report, and the other is presented in Appendix C. Although this latter scheme was not developed under NASA funding, it nevertheless is compatible with the objectives of this contract, and it is therefore fitting to include it here as a proposed topic for further development.

REFERENCES

1. D. C. Gunderson, C. W. Hastings, R. J. Broen, Spaceborne Memory Organization, Proposal to NASA Electronics Research Center, Cambridge, Mass., Honeywell Systems & Research Division, 21 April 1965.

2. D. C. Gunderson, C. W. Hastings, G. J. Prom, Interim Report - Spaceborne Memory Organization, Contract Number NAS 12-38, Honeywell Systems & Research Division, 15 December 1965.

3. D. C. Gunderson, C. W. Hastings, G. J. Prom, Phase I Final Report-Spaceborne Memory Organization, Contract Number NAS 12-38, Honeywell Systems & Research Division, April 1966.

4. D. Goetze, R. Gonzalez, R. D. Gumm, C. W. Hastings, H. R. Holt, J. Kahnke, H. T. Melcher, W. A. Plice, G. J. Prom, D. O. Wick, An Associative Data Handling System for the Small Standard Satellite Program, Proposal to Goddard Space Flight Center, Greenbelt, Maryland, Honeywell Systems & Research Division, 7 September 1966.

5. J. R. Lord, A Proposed Data Compression Study, Proposal, Honeywell Aeronautical Division, 31 January 1964.

6. Data Compression Techniques Study, Proposal to NASA Goddard Space Flight Center, Honeywell Aeronautical Division, 14 May 1964.

7. Analog-Digital Conversion Handbook, Digital Equipment Corporation, Publication No. E-5100, 1964.

8. F. W. Kantor, "Telemetry Concept Could Speed Space Program", Electronics, 12 October 1962, pages 60-62.

9. C. J. Palermo, R. V. Palermo, H. Horwitz, "The Use of Data Omission for Redundancy Removal", technical memo, Institute of Science and Technology, University of Michigan, 1965.

10. R. S. Simpson, C. A. Blackwell, W. O. Frost, "Compendium of Redundancy Removal Processes", IEEE Transactions on Aerospace and Electronic Systems, July 1966, pages 471-474.

11. P. Drapkin, R. M. Pentz, "An Associative Data Compressor", IEEE International Convention Record - Part I, 22-26 March 1965, pages 231-236.

12. D. R. Weber, "A Synopsis on Data Compression", Proceedings of the 1965 National Telemetering Conference, April 1965, pages 9-16.

13. H. N. Massey, "An Experimental Telemetry Data Compressor", Proceedings of the 1965 National Telemetering Conference, April 1965, pages 25-34.

14. W. R. Bechtold, J. E. Medlin, D. R. Weber, Final Report - PCM Telemetry Data Compression Study, Phase I, Contract Number NAS 5-9729, Lockheed Missiles & Space Company, October 1965.

15. L. C. Bunyan, W. P. Hogan, D. R. Weber, Final Report - TIROS Video Data Compression Study, Contract Number NAS 5-9569, Lockheed Missiles & Space Company, December 1965.

APPENDIX A

ANALOG SEARCH TECHNIQUES IN AN ASSOCIATIVE MEMORY

Charles Wm. Hastings

APPENDIX A
ANALOG SEARCH TECHNIQUES IN AN ASSOCIATIVE MEMORY

The procedure described in Subsection III.E of this report for A/D conversion has suggested a novel way of performing very-high-speed searches in an associative memory, which is worth a brief comment here although it is a separate proposal from that of using an associative computer in a data acquisition system.

Figure A1 is a block diagram of an analog search associative memory (ASAM), which can perform equality or inequality searches over relatively short data fields (up to perhaps 10-12 bits) at rates probably exceeding those attainable by any other means; with some modifications which will not be discussed here, it could also perform proximity searches. The principle is very simple: Each strand includes the capability of D/A conversion of one data field, such that the binary contents of that field are also represented by a voltage. If each strand contains more than one field of storage, the other fields might or might not have independent conversion capability. The corresponding field of the search register is similarly available in a voltage representation, which is distributed to each strand. A per-strand two-input comparator is provided, which compares the search voltage with the strand's data field voltage, and produces a logic output according to the result of the comparison - either the search voltage is greater, or the local data field voltage is greater.

A modification of this scheme is necessary to detect <u>equality</u> between the search field and a strand's data field, since two independently generated voltages intended to represent the same binary number would never exactly match, and the comparator would tend to find one of them to be larger. It might be that a more sophisticated comparator circuit could detect an equality condition by determining that the difference between the two voltages was less

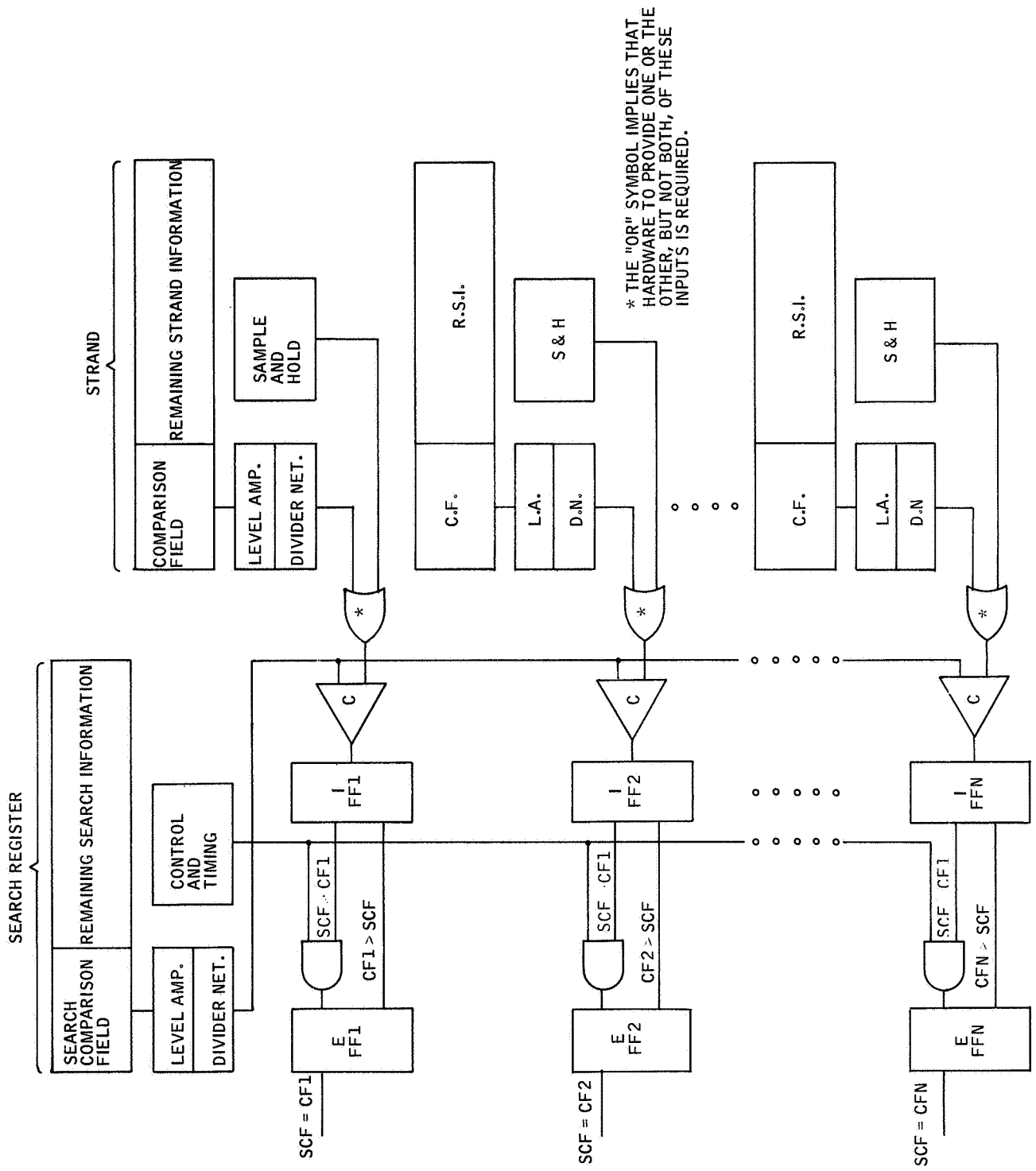


Figure A1. Analog Search Associative Memory

than some threshold. However, the following mode of operation places no special requirements on the comparator and suffices to detect equality: At the beginning of each equality search, the search voltage (corresponding to the nominal values of the search field) is arbitrarily reduced by an amount θ corresponding to one-half of the voltage equivalent of the least significant bit of the search field. The search voltage is then smoothly increased during the search operation, until it reaches a value of θ greater than its normal value. During the search process, the comparator will change state in any strand whose data field nominally matches the search field, since at some time during the search operation the search voltage first equals and then exceeds the strand data field voltage. Hence, equality may be detected by allowing the logic output of a strand's comparator to set one of its flipflops if this output changes during the search. It is assumed here, obviously, that the strand data field voltage is within θ of its nominal value.

The per-strand D/A conversion operation could be done, of course, by equipping each strand with a set of level amplifiers and a ladder network, plus a flipflop register long enough to contain one data field. However, a more economical approach is to provide a special sample-and-hold amplifier per strand, which could retain an impressed voltage for a relatively long time and could be "read nondestructively" through a high-impedance isolation amplifier. D/A conversion could then be centralized to at least some extent. The voltage values stored in the sample-and-hold amplifiers would need to be regenerated periodically, either when enough time had elapsed that the accuracy of the impressed signal could no longer be taken for granted, or else when the ASAM received new information for storage.

The precision limitations of analog techniques are well known. A precision of 10-12 bits wouldprobably require careful design for an ASAM of the type considered here; for one thing, the search voltage must be distributed to every strand of the memory without loss of accuracy. A precision of 12-14 bits is not uncommon nowadays in those high-speed commercial analog multiplexers and A/D converters capable of handling not more than about 50,000

conversions per second. A 20-bit A/D converter has recently been announced by Non-Linear Systems, Inc., Del Mar, California, but it is almost certainly very much slower; ultra-accurate analog equipment has to be slower, since signals must be given much longer to stabilize and certain speedup techniques in A/D converter organization cannot be used which are feasible where less conversion accuracy is called for. Since speed is the major virtue of the ASAM, as contrasted with other associative memory designs, an ASAM would be somewhat inappropriate for applications in which the precision requirements placed on a single data field exceeded the precision of good analog equipment. Nevertheless, it would still be possible to in effect break a long binary datum into "digits" of 8-10 bits each, to use a sample-and-hold amplifier per "digit", and to perform equality and inequality search operations using a kind of analog multiple-precision organization in which one voltage represented the "most significant digit" and so forth.

A classification of approaches to associative system design has previously been defined. * Five approaches are discussed: approaches 1 and 2 are based on reading out a bit slice at a time, as in the proposed adaptive data acquisition system associative computer; and approaches 3, 4, and 5 are based on "local logic" processing, in which each storage element itself also has the capability of determining in some way if its content matches the corresponding search bit. The ASAM would have to be fitted into this classification scheme as "Approach 6", since - unlike any of the others - it is capable of performing an inequality search in effect instantly, without the need for a binary ripple carry or its equivalent to occur. In fact, because of its "threshold" computational organization, an ASAM could probably perform an inequality search more rapidly than an equality search if designed to do both optimally, which is quite a change from the other approaches.

An ASAM could certainly be built with state-of-the-art components, although its precision might be somewhat limited and the performance of the first attempted model might not exceed that attainable with local-logic techniques.

---

*D. C. Gunderson, C. W. Hastings, G. J. Prom, Interim Report - Spaceborne Memory Organization, Contract Number NAS 12-38, Honeywell Systems & Research Division, 15 December 1965.

However, from a longer-range viewpoint there are certain applications, such as checking a table of already detected radio or radar frequencies in an avionic counter-countermeasures system, where the limited precision of an ASAM would be adequate and its great speed would be advantageous. Further improvements in accuracy are to be expected with improvements in analog components, making the ASAM competitive for a broader range of applications.

APPENDIX B

A DESIGN FOR THE EXTERNAL WORD (STRAND) LOGIC OF AN ASSOCIATIVE PROCESSOR

Gregory J. Prom

NOTE: This appendix was prepared as a Honeywell internal memorandum on an Independent Research project in the Computer Technology Group, Systems and Research Division Research Department. It reports on a study very similar to what is planned for the associative computer design being considered on this contract. Although the design approach of this appendix is similar to the one we are taking on the contract, much of this similarity is the result of "convergence" since no deliberate attempt was made to keep the contract work consistent with the Independent Research work or conversely.

APPENDIX B
A DESIGN FOR THE EXTERNAL WORD (STRAND) LOGIC OF AN ASSOCIATIVE PROCESSOR

SUMMARY

The logic design and associated system considerations for the external word (strand) logic in an associative processor are discussed, and a particular design example is presented. Implementation is based on the use of micro-instructions which are basic portions of the usual set of instructions.

INTRODUCTION

The two portions of an associative processor that require the greatest amount of additional study consist of the external word (strand) logic and the control logic. The study reported was conducted to discover some of the potential problem areas in the first of these two parts of an associative processor and to obtain an estimate of the complexity of the resulting logic.

SYSTEM DESIGN PRINCIPLES

A block diagram of the assumed organization for one stage or cell of this logic is shown in Figure B1. It includes a sense amplifier and flip-flop C which are used to read a bit out of the memory and hold it for future use. Flip-flop A holds the primary results of an associative memory operation, such as the sum bit in an addition, for possible further use or for rewriting back into the memory. Flip-flop B holds secondary results, such as the carry bit in an addition, for future use. The D flip-flop contains a mask bit which determines whether or not a particular word (strand) is to be included in a given operation.

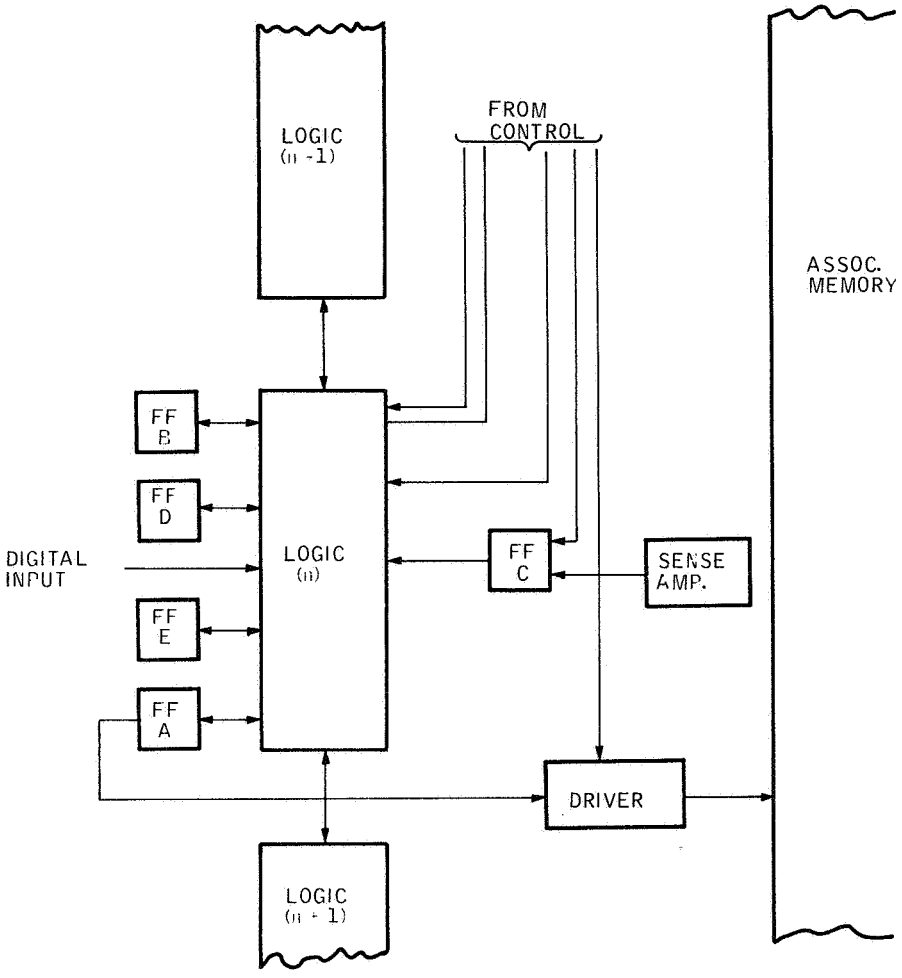


Figure B1. Block Diagram – External Logic

Flip-flop E is used to hold each bit, one at a time, as they are required by the algorithm, in a field multiply operation. The driver accepts data either from an input device or that resulting from an operation and writes it into the memory in conjunction with interrogate drivers.

Initial logic design efforts were based on the concept of providing the complete instruction set as command signals to the logic for each word (strand). This resulted in a very large logic network along with an extremely large number of input (command) leads. To circumvent this problem, it was decided to break down these macro-instructions into a set of micro-instructions, each representing either a simple macro-instruction or a "basic" operation from a complex macro-instruction.

MICRO-INSTRUCTIONS

The resulting list of micro-instructions (those that have been generated to date) is given in Table B1. Not all of these are essential, even for the instructions presently envisioned. Some have been added in order to decrease the execution time of particular instructions by reducing the number of micro-operations per iterative step. Others have been included to facilitate the addition of other macro-instructions. It is, however, a relatively simple matter either to add micro-instructions to the list or remove them. The required logic changes can be made with relative ease because of the form of the mechanization.

MACRO-INSTRUCTIONS

A set of macro-instructions is given in Table B2. Only those instructions that require the use of two or more micro-instructions have been included. However, some of the micro-instructions are useful in their own right and should therefore be considered part of the instruction list for the associative processor. Several of the instructions in this list have been broken down into micro-instruction

Table B1, Micro-Instructions

No.	Instruction
1	$B \oplus C \rightarrow A$
2	$A \oplus C \rightarrow A$
3	$A \oplus S \rightarrow A$
4	$BC \rightarrow B$
5	$B + AC \rightarrow B$
6	$B + AS \rightarrow B$
7	$A\ (\overline{C \oplus S}) \rightarrow B$
8	$B\ (\overline{\bar{A}\bar{S}C}) \rightarrow B$
9	$\bar{A} \rightarrow A$
10	$A \rightarrow A$
11	$B \rightarrow A$
12	$0 \rightarrow A$
13	$I \rightarrow A$
14	$A_{n-1} \rightarrow A_n$
15	$A_{n+1} \rightarrow A_n$
16	$A \oplus B \rightarrow A$
17	$A + B \rightarrow A$
18	$AB \rightarrow A$
19	$\bar{C} \rightarrow C$
20	$A_n \rightarrow B_n$ iff $A_{n-1} = A_{n-2}$ ------- $= A_1 = A_0 = 0$ (First "1" in $A \rightarrow B$)
21	$0 \rightarrow A_n$ if $B_n = 1$
22	$0 \rightarrow B$
23	$C \rightarrow D$
24	Disable (1, 2, and 3) and $C \rightarrow A$ if $D = 0$
25	Disable J_A and K_A Logic if $E = 0$
26	$A \rightarrow D$
27	(Input) $\rightarrow A$
28	Set FF in control section iff $A_1 + A_2 +$ -------$+ A_n = 1$ (Booleansum)

Table B2. Macro-Instructions

No.	Instruction	Macro-Instructions	
		Initialization	Iterative Subroutine
1	ADD	(22)	(1, 4)* (2, 5)
2	FIELD ADD	(22)	(1, 4) (3, 6)
3	SUB	(13, 10)	(19, 1, 4) (3, 6)
4	FIELD SUB	(13, 10)	(1, 4) (19, 2, 5)
5	EQUALITY SEARCH	(13)	(7)
6	INEQUAL SEARCH	(13, 10)	(7, 8)
7	MAXIMUM SEARCH	(13, 10)	(7) (28) (10) (11)
8	MINIMUM SEARCH	(12, 10)	(7) (28) (10) (11)
9	LOCATE FIRST MATCH	(22)	(9)
10	STORE A		(WRITE)
11	STORE INPUT	(27)	(WRITE)
12	MULTIPLY		
13	FIELD MULT.		
14	DIVIDE		
15	FIELD DIV.		
16	LEFT SHIFT		
17	RIGHT SHIFT		

*Numbers within parentheses designate micro-instructions performed simultaneously within one computer step.

sequences. Although an equivalent breakdown has not been completed for the remaining instructions, throught has been given to their mechanization, particularly in the selection of micro-instructions. Note that each instruction is executed by first performing certain initializing operations, after which an iterative subroutine is repeated, once for each bit in the word.

LOGIC DESIGN

A set of logic equations for implementing the micro-instruction set of Table B1 is given in Table B3. These equations were developed for J-K flip-flops since this type appears to be the most common type available in high-speed micro-circuit form.

Table B3. Logic Equations

$$J_A = \left\{\left[(\overline{24}) + D\right]\left[(B \oplus C)\,(1) + C\,(2) + S\,(3)\right] + (9) + (13) + B\,(11 + 16 + 17) + A_{n-1}\,(14) + A_{n+1}\,(15) + \overline{CD}\,(24) + (\text{INPUT})\,(27)\right\}\,(\overline{27} + E)$$

$$K_A = \left\{\left[(\overline{24}) + D\right]\left[(B \oplus \overline{C})\,(1) + C\,(2) + S\,(3)\right] + (C \oplus S)\,(7) + (9) + (12) + \bar{B}\,(11 + 18) + B\,(16 + 21) + \bar{A}_{n-1}\,(14) + \bar{A}_{n+1}\,(15) + \overline{CD}\,(24) + \overline{(\text{INPUT})}\,(27)\right\}\,(\overline{25} + E)$$

$$J_B = AC\,(5) + AS\,(6) + A\,(10) + P\,(20)$$

$$K_B = \bar{C}\,(4) + \overline{ASC}\,(8) + \bar{A}\,(10) + \bar{P}\,(20) + 22$$

$$J_C = K_C = 19$$

$$J_D = C\,(23) + A\,(26) \qquad K_D = \bar{C}\,(23) + \bar{A}\,(26)$$

A mechanization, in terms of NAND gates and J-K flip-flops is shown in Figure B2. A particular J-K flip-flop manufactured by Sylvania was assumed. It has a small amount of internal logic, associated with each input, which results in a slightly reduced external logic requirement. Nevertheless, a total of about 20 integrated-circuit flat packs are required based on presently available packages. It should be possible to apply LSI (large-scale integration) techniques to this circuit, however, and thereby place the entire logic circuit (including flip-flops, sense amplifier, and possibly drivers) on a single chip. With this in mind, the circuit was designed to be as universal as possible, even at the expense of additional logic complexity.

As an example, the pointer logic, which scans the results register (A) for 1's and indicates the location of the first 1 in the register, was designed to permit each stage of the logic to be identical to every other stage and at the same time to speed the carry time as much as is possible under this constraint. To accomplish this, a modified multicell iterative circuit technique, with resulting carry speedup, was used even though 4 NANDS with a total of 12 inputs are required. For SUHL II logic (by Sylvania) a propagation speed of about 1.7 nanoseconds per word (strand) results. Another technique, which has yet to be tried with present day transistors, is shown in Figure B3. This is a conventional iterative contact network with transistor switches replacing the relay contacts. Only three transistors per stage are required, but several stages would probably have to be placed on a single chip, if it was to be integrated, rather than being portions of many chips. The speed of this circuit will be quite high, but the actual figures will not be known until laboratory tests can be run on the circuits. Delay times of about 5 nsec per stage were obtained about 8 years ago using lower speed germanium transistors in the same basic circuit.

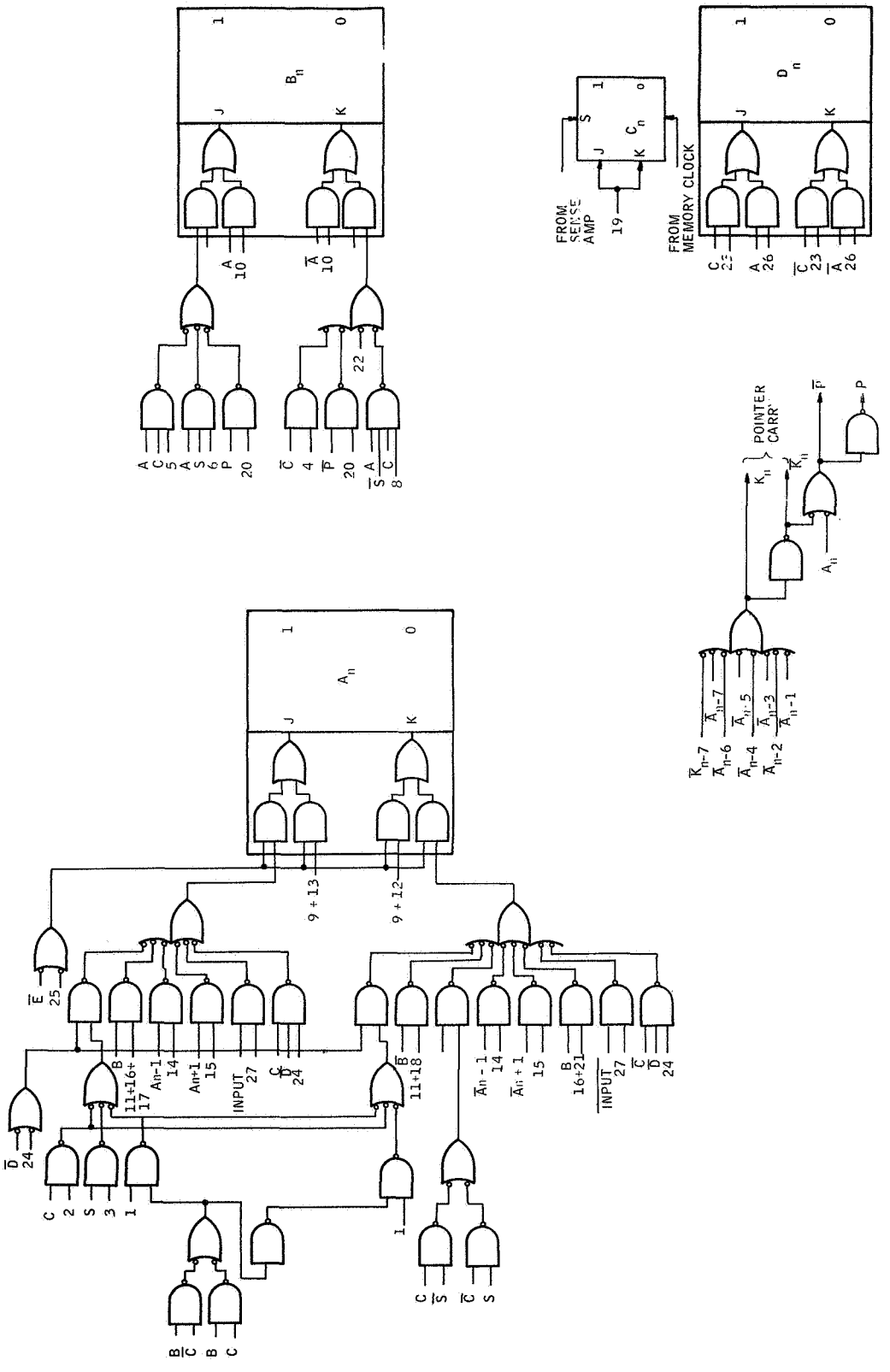


Figure B2. Logic Design Mechanization

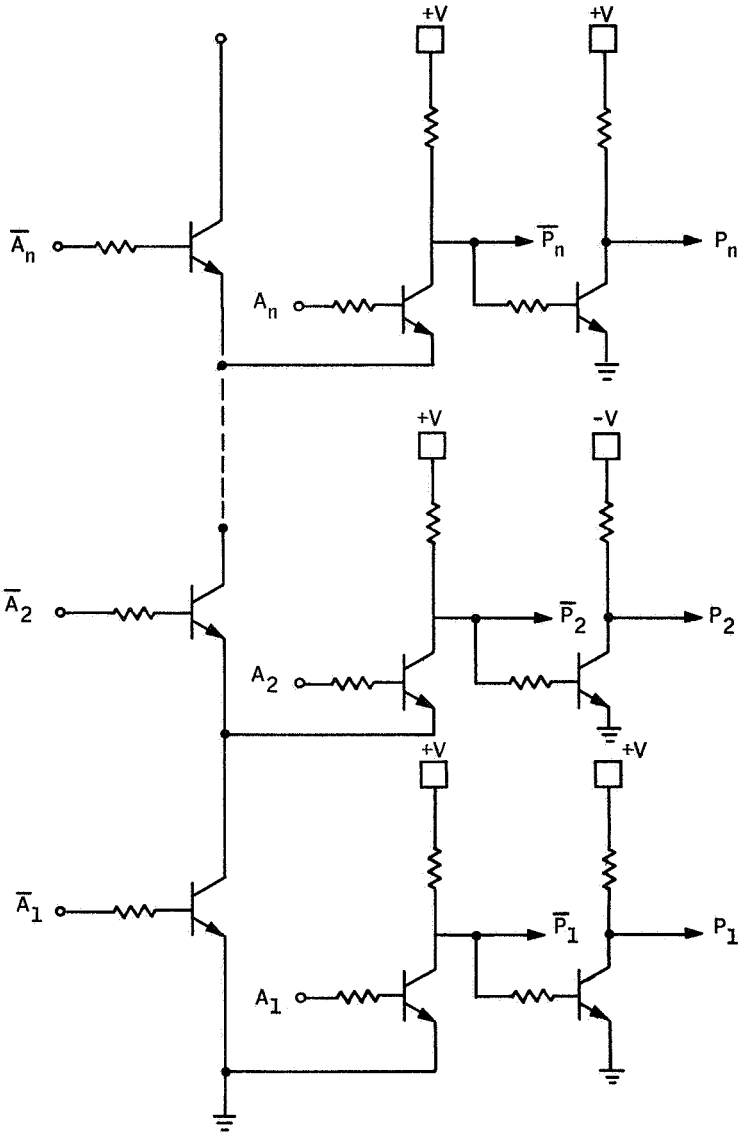


Figure B3. "Pointer" Logic

CONCLUSIONS

The preceding instruction lists, logic equations, and logic diagrams have been developed in an attempt to get a feel for the problems involved in the design of an associative processor without local logic and to obtain an estimate of the complexity of the external logic. The results indicate that a considerable amount of logic is required, particularly if the goals of high operating speeds, flexibility, large number of instructions, and adaptability to LSI techniques are desired. Admittedly, there are many applications in which a simpler logic circuit implementing a much smaller set of instructions is adequate.

It was found, for the set of instructions considered in this example, that breaking the instructions down into more basic operations termed micro-instructions resulted in a simpler logic circuit than that resulting from a straightforward mechanization of the original instruction set. As the number of instructions decreases, and particularly the number of similar instructions, this conclusion may no longer be accurate.

APPENDIX C

PULSE-TYPE ANALOG-TO-DIGITAL CONVERSION

William A. Plice

APPENDIX C
PULSE-TYPE ANALOG-TO-DIGITAL CONVERSION

Note: This appendix describes a novel analog-to-digital conversion technique. It was prepared as a Honeywell internal memorandum by William A. Plice of the Honeywell Aeronautical Division's Aerospace Support Equipment Group. The idea was conceived and the work was performed by Mr. Plice, on Honeywell independent development funds.

The analog-to-digital conversion technique described here is of the "sectioned-counter" type, and would have the same speed as the other sectioned-counter scheme (see Subsection III-E of this report) for a given allocation of counter bits into fields. As contrasted with the other scheme, this one requires fewer linear circuits; however, those linear circuits which are required would probably need to be somewhat more precise than with the other scheme, to achieve the same conversion accuracy.

SUMMARY

(1) One unusual technique for A/D conversion is described.

(2) This method appears attractive where data must be converted from many sources simultaneously.

BACKGROUND

Most electrical analog-to-digital converters operate upon the principle of comparing an unknown voltage to a known one. The known voltage is directly proportional to and controlled by a digital number. This number is changed according to some strategy until the known and unknown are equal to the desired degree of accuracy. The digital number at the time of equality is then equal to the unknown multiplied by some calibration factor.

In many cases it is desirable to simultaneously convert many unknown voltages. The technique herein described is suitable for this situation and is accomplished with a relatively small amount of hardware. The description to follow is based upon conversion by groups of three binary digits (octal representation). This is arbitrary, and groups of two, four, or more bits may be used if desired.

For each signal source to be converted, an integrator, a comparator, a flipflop, and miscellaneous control and switching circuitry is provided. A sketch is shown in Figure C1.

The signal to be converted is integrated for a fixed period of time, producing an output

$$e_o = \int_0^T \frac{e_x(t)}{RC}\, dt.$$

For purposes of discussion we may assumes that $e_x(t)$ is substantially constant over the interval T so that $e_o \approx e_x T/RC$.

The conversion is begun with all bits in a binary counter set to one. In octal, then, the number in the counter is initially 7777. The most significant three bits are then allowed to count in a binary fashion from 111 to 000 to 001, and so forth, back to 111. For each count a pulse of current of fixed amplitude and duration is sent to the integrator. The pulse amplitude and width are adjusted so that eight of these pulses will when summed in the integrator, be equal and opposite to the effect of a full-scale input voltage.

Since the voltage applied to the various signal channels will, in general, not be greater than or equal to full scale, as the count progresses the integrator of one or more channels will change sign on some particular count. This occurrence is sensed by the comparator and the flipflop is set. Then the state of the first three bits is transferred to the corresponding bits of a register associated with each of the integrators that have changed state on that count. Further integration is inhibited by the set flipflop.

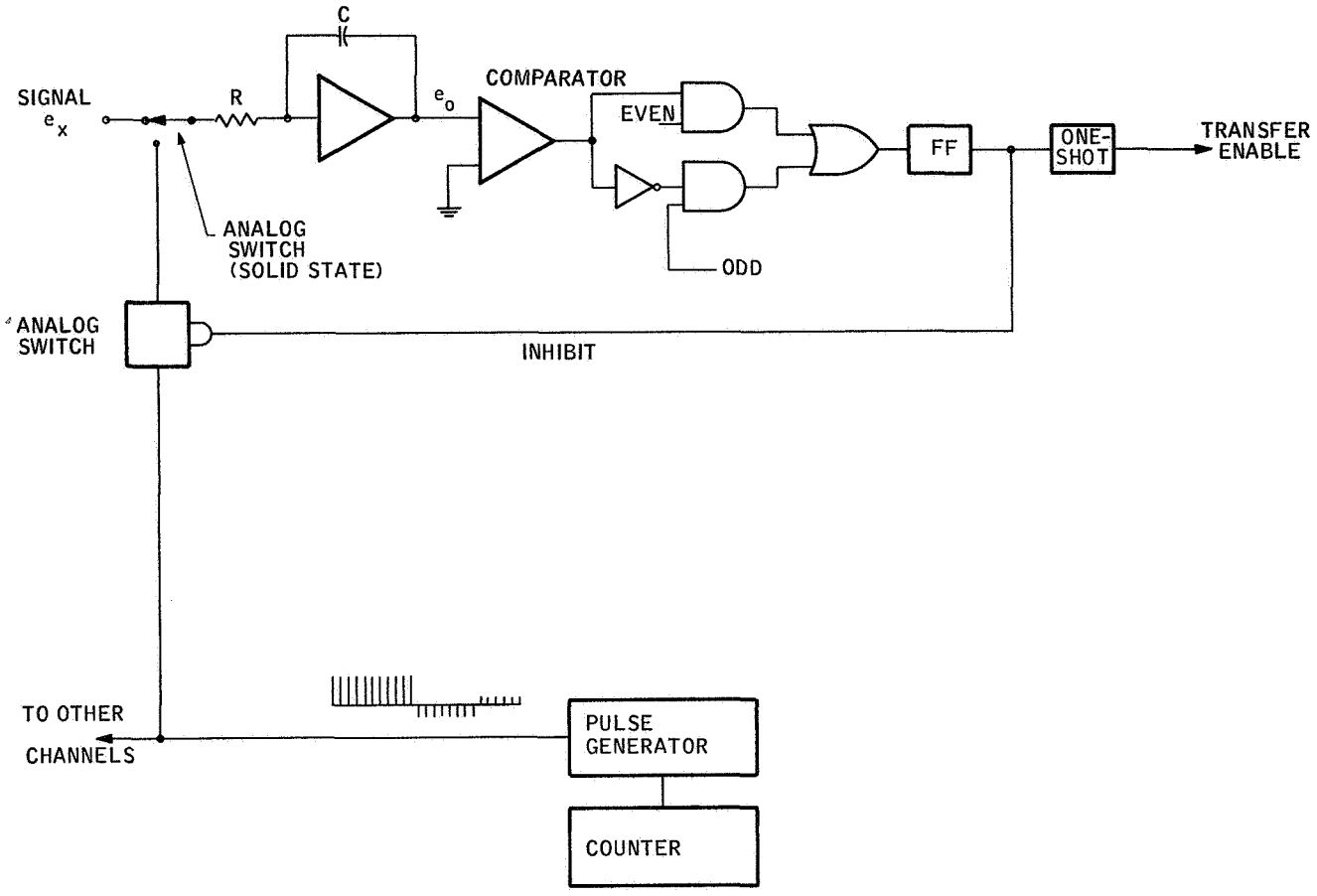


Figure C1. Pulse-Type Sectioned-Counter A/D Converter

Since the count began at 111, the number transferred to the register is, in binary notation, one less than the actual number of pulses emitted. The voltage of the integrator may then be expressed as

$$e_o = e_x \frac{T}{RC} - \frac{(n_1 + 1)}{8} \cdot E_F \frac{T}{RC}$$

where E_F is the full scale voltage and n_1 is the number transferred to the register. After the first three bits have made a complete cycle to 111, the flipflops are reset and the next most significant three bits allowed to count. For each of these counts a pulse of 1/8 the weight of the previous pulses is emitted. This may be accomplished by reducing amplitude, time, or both. These pulses are in the same direction as the original signal.

The rule for conversion starting with most significant group first is:

(1) Odd Groups

Use pulses opposing signal of weight 2^{-n} where n is the bit position of the least significant position in the group. Transfer C-1 where C is the number of counts required to cause the comparator to change state.

(2) Even Groups

Use pulses aiding the signal of weight 2^{-n} where n is the bit position of the least significant position in the group. Transfer 2^b-C where C is the number of counts required to cause the comparator to change state, and b is the number of bits in the group.

When this rule is followed for groups of three bits each, the number obtained is $[C_1-1]\,[8-C_2]\,[C_3-1]\,[8-C_4]$ where C_x is an octal number and the integrator output is

$$e_o = e_x \frac{T}{RC} - \frac{C_1}{8} \frac{E_F T}{RC} + \frac{C_2}{64} \frac{E_F T}{RC} - \frac{C_3}{512} \frac{E_F T}{RC} + \frac{C_4}{4096} \frac{E_F T}{RC}$$

Now if the conversion is complete $e_o \approx 0$ and

$$e_x = E_F \left[\frac{C_1}{8} - \frac{C_2}{64} + \frac{C_3}{512} - \frac{C_4}{4096} \right]$$

but this is equivalent to:

$$e_x = E_F \left[\frac{C_1 - 1}{8} + \frac{8 - C_2}{64} + \frac{C_3 - 1}{512} + \frac{8 - C_4}{4096} \right]$$

Thus, the number transferred to the holding register is a true representation of the actual voltage, since the quantity in brackets represents a digital number in octal notation equal to that transferred to the holding register.

In practice C-1 is formed by setting the counter group to 111 and counting up. 8 - C is formed by presetting to 111, counting up, and then transferring the 1's complement of the final state.

It should be apparent that the counter can be grouped in any desired manner, and that in particular 4-bit BCD can be obtained.

The diagram and explanation are very brief, but all elements are within the state-of-the-art.

The advantages of this scheme are:

(1) Simultaneous conversion of many signals.

(2) Sample-and-hold capability.

(3) Minimum analog hardware.

(4) Relatively fast conversion.

(5) The value of C is not critical, nor is R if it is used both for signal and for reference pulses. Only a few precision resistors are required.

(6) High frequency noise is averaged out.